On Modeling and Assessing Uncertainty Estimates in Neural Learning Systems

Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt
von
Joachim Sicking
aus
Coesfeld

Bonn, 2022

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter: Prof. Dr. Stefan Wrobel
2. Gutachter: Prof. Dr. Christian Bauckhage

Tag der Promotion: 14.04.2023
Erscheinungsjahr: 2023

Betreuung der Dissertation am Fraunhofer IAIS:
Dr. Maram Akila, Dr. Tim Wirtz

Acknowledgments

One of the most enjoyable aspects of doing research is developing and discussing ideas with colleagues, preferably while standing in front of a whiteboard or near a coffee machine. The hints, sketches, and new perspectives generated in these talks help research activities to hatch and make them grow. Thank you all for this support.

Some folks deserve special mention: first, Maram Akila, who supervised large parts of this thesis. His curiosity, sharpness and serenity were of great help more than once. Whether it is searching for a difficult-to-spot research path, doing something mundane like debugging code, or discussing *Rick and Morty* episodes—Maram tends to contribute thoughts of surprising clarity. Further thanks go to Tim Wirtz, who also supervised me, for his "always-on" motivation and for being the idea-generation machine he is. I would also like to mention Max Pintz, who developed from a Bachelor's student to a PhD student during the course of this thesis. Among many other contributions, he tamed the previously wild `DenseHMM` code base just at the right moment. Not to forget Hojun Lim, who joined our institute as a Master's student during COVID-19. While the pandemic tried to reduce him to a rectangular pane in a communication tool, he fiercely resisted and proved that remote work can be a pleasant and rewarding experience.

Getting a bit more formal, special thanks go to my professorial supervisor Stefan Wrobel for his guidance and to Christian Bauckhage for serving as the second examiner. I would further like to acknowledge Alexander Kister for introducing me to the topic of uncertainty estimation, Asja Fischer for sharpening some of our research contributions, and Daniel Trabold for proofreading this thesis. Thanks also for the productive work to my other collaborators and co-authors over the years, namely, Stefan Eickeler, Matthias Fahrland, Sebastian Houben, Stefan Rüping, Emanuel Müller, and Jasmin Brandt.

This team "research NRW" is complemented by the team "industry Lower Saxony", i.e., Peter Schlicht, Fabian Hüger and Jan David Schneider from Volkswagen. Our projects on uncertainty estimation for autonomous driving kick-started several research activities and contributed to this work by providing a well-informed practitioners' perspective. The same applies to the publicly sponsored project "KI-Absicherung - Safe AI for Automated Driving", which gratefully provided funding for my research activities, as did the Fraunhofer Center for Machine Learning. Furthermore, it is helpful to have supportive project leads and department heads. Thanks in this respect go to Dirk Hecker, Stefan Rüping, Tim Wirtz, Maximilian Poretschkin, Michael Mock, and Sebastian Houben.

Widening the focus, I would like to say a warm thank you to Sven Giesselbach for cooking evenings, good talks, and a memorable conference trip to North Macedonia in 2017. Moreover, special thanks to my office mates David Gutmann (at that time, Knodt), Julia Rosenzweig, and Sujan Sai Gannamaneni for all our small conversations, the enlightening, and the fun ones. Finally, I am glad to have awesome companions for adventures and enjoyments in life beyond scientific research. These are my friends, my parents and siblings, and, of course, my girlfriend Maria. You rock.

Abstract

While neural networks are universal function approximators when looked at from a theoretical perspective, we face, in practice, model size constraints and highly sparse data samples from open-world contexts. These limitations of models and data introduce uncertainty, i.e., they render it unclear whether a model's output for a given input datapoint can be relied on. This lack of information hinders the use of learned models in critical applications, as unrecognized erroneous predictions may occur. A promising safeguard against such failures is *uncertainty estimation*, which seeks to measure a model's input-dependent reliability. Theory, modeling, and operationalization of uncertainty techniques are, however, often studied in isolation. In this work, we combine these perspectives to enable the effective use of uncertainty estimators in practice. In particular, it is necessary to address (the interplay of) three points. First, we need to better understand the theoretical properties of uncertainty estimators, specifically, their shortcomings stemming from constrained model capacity. Second, we must find a way to closely model data and error distributions that are not explicitly given. Third, for real-world use cases, we need a deeper understanding of uncertainty estimation requirements and their test-based evaluations.

Regarding the first point, we study how the estimation of uncertainty is affected (and limited) by a learning system's capacity. Beginning with a simple model for uncertain dynamics, a hidden Markov model, we integrate (neural) word2vec-inspired representation learning into it to control its model complexity more directly and, as a result, identify two regimes of differing model quality. Expanding this analysis on model capacity to fully neural models, we investigate Monte Carlo (MC) dropout, which adds complexity control and uncertainty by randomly dropping neurons. In particular, we analyze the different types of output distributions this procedure can induce. While it is commonly assumed that output distributions can be treated as Gaussians, we show by explicit construction that wider tails can occur.

As to the second point, we borrow ideas from MC dropout and construct a novel uncertainty technique for regression tasks: *Wasserstein dropout*. It captures heteroscedastic aleatoric uncertainty by input-dependent matchings of model output and data distributions, while preserving the beneficial properties of MC dropout. An extensive empirical analysis shows that Wasserstein dropout outperforms various state-of-the-art methods regarding uncertainty quality, both on vanilla test data and under distributional shifts. It can also be used for critical tasks like object detection for autonomous driving. Moreover, we extend uncertainty assessment beyond distribution-averaged metrics and measure the quality of uncertainty estimation in worst-case scenarios.

To address the third point, we need not only granular evaluations but also have to consider the context of the intended machine learning use case. To this end, we propose a framework that i) structures and shapes application requirements, ii) guides the selection of a suitable uncertainty estimation method and iii) provides systematic test strategies that validate this choice. The proposed strategies are data-driven and range from general tests to identify capacity issues to specific ones to validate heteroscedastic calibration or risks stemming from worst- or rare-case scenarios.

List of Publications

Parts of this thesis have already been published:

- J. Sicking, M. Pintz, M. Akila, T. Wirtz, *DenseHMM: Learning HMMs by Learning Dense Representations*, International Conference on Pattern Recognition Applications and Methods 2022, DOI: 10.5220/0010821800003122
- J. Sicking, M. Akila, M. Pintz, T. Wirtz, S. Wrobel, A. Fischer, *Wasserstein Dropout*, Machine Learning (Springer), DOI: 10.1007/s10994-022-06230-8
- J. Sicking, M. Akila, D. Schneider, F. Hüger, P. Schlicht, T. Wirtz, S. Wrobel, *Tailored Uncertainty Estimation for Deep Learning Systems*, under review, pre-print on arXiv: 2204.13963
- J. Sicking, M. Akila, T. Wirtz, S. Houben, A. Fischer, *Characteristics of Monte Carlo Dropout in Wide Neural Networks*, Workshop on Uncertainty and Robustness in Deep Learning at ICML 2020
- J. Sicking, A. Kister, M. Fahrland, S. Eickeler, F. Hüger, S. Rüping, P. Schlicht, T. Wirtz, *Approaching Neural Network Uncertainty Realism*, Workshop on Machine Learning for Autonomous Driving at NeurIPS 2019
- J. Sicking, M. Akila, M. Pintz, T. Wirtz, A. Fischer, S. Wrobel, *A Novel Regression Loss for Non-Parametric Uncertainty Optimization*, 3rd Symposium on Advances in Approximate Bayesian Inference 2021

Two Master's theses were written in the context of this dissertation:

- M. Pintz, *Novel Optimization Schemes for Uncertainty Estimation in Regression Neural Networks*, University of Bonn, Computer Science, June 2021
- H. Lim, *Improved Uncertainty Estimation in Object Detection Networks*, RWTH Aachen, Media Informatics, October 2021

List of Code Repositories

Some of the publications listed above are accompanied by publicly available source code repositories that allow for reproducing most of the numerical results presented in this thesis:

- *DenseHMM: Learning HMMs by Learning Dense Representations*:
  `https://github.com/fraunhofer-iais/dense-hmm`
- *Wasserstein Dropout*:
  `https://github.com/fraunhofer-iais/wasserstein-dropout`
- *A Novel Regression Loss for Non-Parametric Uncertainty Optimization*:
  `https://github.com/fraunhofer-iais/second-moment-loss`

Contents

1. Introduction

Having attracted great attention in both academia and the digital economy, machine learning (ML) models are about to become vital components of safety-critical applications. Examples include autonomous driving (Bojarski et al., 2016; Zeng et al., 2019) and medical diagnostics (Liu et al., 2014; Zhou et al., 2019), where unrecognized prediction errors potentially put humans at risk. These systems require methods that are reliable not only on average and under clean-room conditions, e.g., when only controlled and curated inputs are being presented, but also in the messy and unforeseeable outside world that is characterized, e.g., by continuous data shifts or the need to explicitly consider worst-case scenarios. Autonomous driving (AD) systems, for instance, are employed in open-world contexts and are thus likely to encounter challenging and structurally new scenarios during operations (see Fig. 1.1). While such potentially critical situations are (fortunately) rare, they may significantly impact the safety of operations, as it is typically unclear whether systems learned from data can be successfully applied to such novel scenarios, i.e., whether they *generalize* well. This holds in particular for the widely used model class of *neural networks* (NNs), (non-)linear nestings of (sometimes millions or billions of) atomic units termed *neurons*. These non-interpretable black-box models provide unprecedented performance on a broad range of ML tasks; however, their generalization abilities are notoriously difficult to predict. Cases where a lack of reliability even led to fatal accidents are well documented for NN-based autonomous driving systems (see, e.g., NTSB (2019a) and NTSB (2019b)).

Trustworthy machine learning, which is also often called *safe machine learning*, is a sub-field of ML that addresses such model insufficiencies that prevent recent ML models, despite their potential, from being broadly deployed in safety-critical applications (see, e.g., Amodei et al. (2016); Koopman (2018); Houben et al. (2022)). Safe-ML techniques for higher reliability[1] are motivated by the observation that classical approaches to improving model quality, which often build on collecting and employing ever larger and more representative datasets, are subject to limitations. While enormous training sets may help reduce a model's uncertainty (Vapnik, 2000) and thus the need to accurately model it, we can still not hope to "scale away" uncertainty given the sheer size of many model input and output spaces and irreducible data noise. Instead, modeling techniques are required that reflect uncertainty, for instance, by generating a distribution over plausible outputs (for a given input) instead of predicting a seemingly unambiguous point estimate. Such an output distribution should reflect modeling insufficiencies (e.g., due to model size constraints and sparse data samples), and, additionally, grasp intrinsic data ambiguity

[1] Not all safe-ML techniques tackle performance-related issues and may instead concern, e.g., fairness or interpretability. In this work, however, we focus on prediction quality.

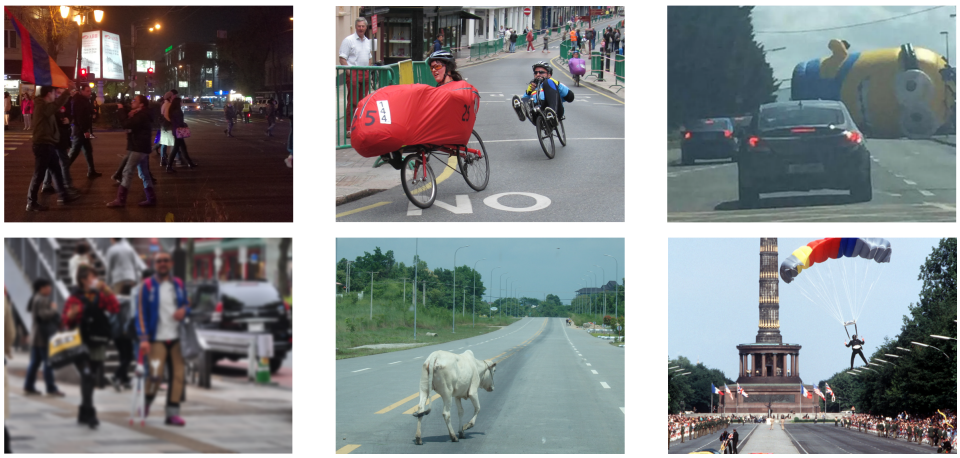


Fig. 1.1.: Examples of challenging and structurally new scenarios that an autonomous driving (AD) system may encounter during operations and that might cause insufficient or erroneous model behavior. The ability to detect such scenarios is crucial for safe AD, as it allows for activating redundant algorithmic systems or initiating a reduction of the vehicle speed. The displayed scenes differ in their probability of occurrence: while low contrasts and blurry sensor information (see left column) can be expected, not all types of traffic participants (see middle column) can be foreseen. Anticipating concepts that are unrelated to land transportation (like inflatable objects and parachutes, see right column) is almost impossible. The images are taken from various sources; see Challenging Traffic Scenes (2022).

(e.g., due to the stochastic nature of a system or measurement noise). These model and data limitations are commonly referred to as *epistemic uncertainty* (also called model uncertainty) and *aleatoric*[2] *uncertainty* (also termed data-inherent uncertainty). For a detailed discussion of these concepts, see Section 3.1. Here, we illustrate them through two examples:

In Fig. 1.2, manual segmentations[3] of human tissue by four radiologists are considered, which can be used to learn ML models for medical diagnosis. This semantic classification of pixels as "inconspicuous" or "conspicuous" is subject to data-intrinsic uncertainty (aleatoric uncertainty) due to the blurriness of the computed tomography (CT) image and due to (resulting) differences between the experts' assessments. Their respective image annotations are visualized in two different ways (see the l.h.s. and r.h.s. of Fig. 1.2). In some cases, the differences among graders are significant, particularly in the last column of the l.h.s. image (highlighted by a red frame) where the assessment of the second radiologist (who classifies the tissue as "inconspicuous") strongly deviates from those of her colleagues. While the availability of such multi-annotations is beneficial, as it allows for approximating

[2] *Alea* is Latin for dice. The intrinsically stochastic dice roll lends its name to this type of uncertainty.
[3] Segmentation is the task of assigning pixel-level (semantic) class labels to an image.

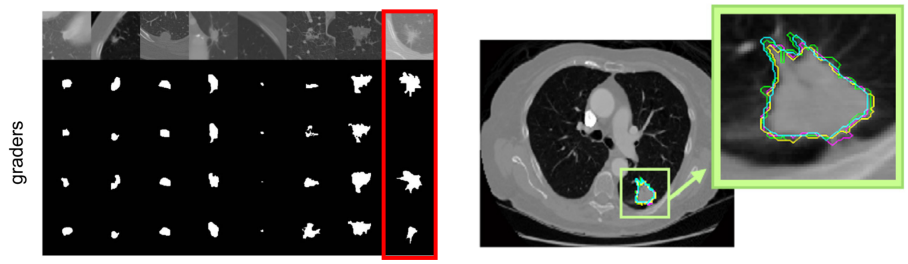


Fig. 1.2.: Illustration of aleatoric uncertainty via annotated lung CT images from the LIDC-IDRI dataset (Armato III et al., 2011). Unlike many other ML datasets, these images come along with multiple annotations and thus provide rough approximations of *uncertainty ground truth distributions*. Specifically, the annotations indicate lung lesions that were marked by four radiologists. Their labelings are shown as segmentation masks (l.h.s., second row to last row) and colored contours (r.h.s.), respectively. The image on the l.h.s. is taken from Kohl et al. (2018) with modifications, the one on the r.h.s. is from Nurfauzi et al. (2021).

uncertainty ground truth distributions, such uncertainty information is not available for the vast majority of ML datasets, rendering uncertainty estimation challenging.

While aleatoric uncertainty is due to the ambiguity of given data, epistemic uncertainty is caused by the sparsity of data samples, which is particularly due to the high dimensionality of input and output spaces. We exemplify epistemic uncertainty with a probabilistic (neural) object detector (see Fig. 1.3) that was designed to capture this type of uncertainty. Being trained to recognize traffic participants, such as pedestrians, cyclists, and cars, "novel" (semantic) concepts in test inputs like a car with an open trunk lid (see the largest detection on the l.h.s. image in Fig. 1.3) or a mannequin (see the detection on the r.h.s. image in Fig. 1.3) cause high model weight uncertainty and thus high predictive uncertainty (see figure caption for details). If the object detector is part of an autonomous driving pipeline, such a lack of knowledge about the exact numbers and positions of traffic participants in a scene may trigger downstream safety or mitigation mechanisms, for instance, more conservative motion planning.

In reality, we typically observe mixtures of aleatoric and epistemic uncertainty. This calls for uncertainty mechanisms that capture both these types. Established and theoretically well-understood (Bayesian) techniques for uncertainty quantification (see, e.g., Section 3.3 of Bishop (2006)), however, do not readily scale to complex (particularly neural) ML models. Other techniques are more scalable, but come at the cost of strongly simplifying assumptions. For classification tasks, one may, for instance, rely on basic scores like softmax statistics (e.g., its largest value (Hendrycks & Gimpel, 2017) or its entropy (Kendall & Gal, 2017)) that come "free of charge" as part of standard modeling approaches.[4] However, while these scores can be interpreted as "ad-hoc" uncertainty estimates, they are typically

[4]For regression tasks, particularly when modeled by standard neural networks, no such "default" uncertainty measures are available.

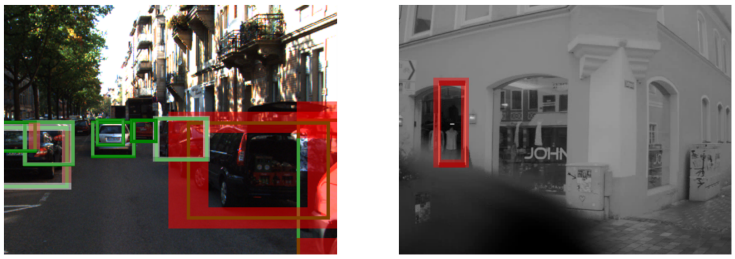


Fig. 1.3.: Visualization of (mostly) epistemic uncertainties in object detection. Each detected "traffic participant" corresponds to a rectangular frame. The thickness and color of the frame encode the respective position uncertainty, where green, thin frames indicate low uncertainty and red, thick frames imply high uncertainty. On the l.h.s., the detection of a dark car (the largest frame) is highly uncertain, likely because of the "novel" concept of an open trunk lid. On the r.h.s., the model confuses a mannequin with a pedestrian. Again, a high uncertainty estimate indicates the model's lack of confidence. The left image is taken from the KITTI dataset (Geiger et al., 2012), with modifications from Pintz (2021), and the right one from the A2D2 dataset (Geyer et al., 2020), with modifications.

ill-calibrated, and calibrating[5] them requires rather fragile additional mappings that often do not generalize well (see, e.g., Snoek et al. (2019)). These observations foster the need for more sophisticated (neural) uncertainty mechanisms that are theoretically well motivated and practically applicable in industry-scale ML systems. A promising and widely used candidate approach in this regard is Monte Carlo (MC) dropout (Gal & Ghahramani, 2016a). Its conceptual simplicity from a practitioner's perspective is, however, contrasted by more involved inner workings (see Appendix of Gal & Ghahramani (2016a)) that cause the theoretical understanding of this Bayesian approximation to be not comprehensive yet, rendering unforeseeable behaviors and error modes more likely. And while a variety of alternative uncertainty approaches have been established (Maddox et al., 2019; Malinin et al., 2020; Amini et al., 2020; Barber et al., 2021; Durasov et al., 2021; Fan et al., 2021), stable quantification of uncertainty is still an open problem. It is, among others, complicated by the fact that uncertainty mechanisms are (in many cases) learned as integrated parts of a model, i.e., they are supposed to quantify modeling insufficiencies resulting from capacity limitations they are themselves subject to. This raises the question of whether or how strongly the meta-task of uncertainty quantification is affected by model capacity constraints and how the quality of uncertainty estimates changes when model capacity is varied.

Apart from well-understood uncertainty estimation techniques, convincing evaluation schemes (see Snoek et al. (2019) for a positive example) are required that facilitate

[5] Probabilistic(ally interpreted) model predictions are called *(well-)calibrated* if they coincide with actually measured statistics. Assignments of images to image categories that are predicted to be 90% confident, for instance, are properly calibrated if these assignments are actually correct in 90% of all cases (when considering a large number of images).

a thorough assessment of the obtained uncertainty estimates. The general absence of uncertainty labels, for instance, does not only complicate modeling uncertainties but also constrains their evaluation. It necessitates either aggregating set-based evaluation metrics (e.g., calibration measures like the expected calibration error, see Section 3.3 for details) that do not allow for assessing the quality of individual uncertainty estimates or, alternatively, performance-biased measures (like the negative log-likelihood (NLL), see ibid. for details) that enable pointwise evaluations but reflect distributional properties only in a very limited fashion. While these measures are statistically well founded (the NLL, for example, is a so-called proper scoring rule (Gneiting & Raftery, 2007)), they are not designed to address the (various types of) safety requirements a real-world ML system is subject to.

From a safety perspective, one may ask for additional measures that resolve uncertainty quality at a more granular level and provide (sufficient) sensitivity in modeling regimes that are identified as potentially critical, for instance, in the case of an AD system, when pedestrians are in the immediate vicinity of the vehicle. Additionally, one may consider risk weightings of quantities that are typically treated on an equal footing by standard measures, for instance, over- and underestimated uncertainties: the former ones likely trigger a more conservative system behavior (causing a delayed arrival but no substantial risks), as an input scenario is (falsely) "perceived" as more challenging than it actually is. The latter, on the contrary, lead to overconfident model outputs that may cause potentially harmful subsequent "actions" that do not adequately reflect the model's actual ability to handle the current input. Uncertainty metrics aside, standard evaluation schemes are, moreover, rather stylized, as they assume, e.g., only discrete domain shifts or consider solely average model performances while largely neglecting reliability in, for example, worst-case scenarios. Such coarse uncertainty evaluations are particularly problematic, as *stability*, understood as maintaining high quality on individual inputs across a wide range of datasets and scenarios, is an important characteristic for mechanisms that are supposed to foster a system's trustworthiness. Depending on the application context, it may even be acceptable to choose a more stable uncertainty estimator if it comes at the cost of (slightly) reduced uncertainty quality in best-case scenarios.

To evaluate such balances between uncertainty properties, systematic testing is required. Looking more broadly at our abilities to check such properties, we notice, however, that established safety-critical systems are white boxes. They are manually composed, and knowledge on physics, materials science, or code frameworks, for example, allows for an (at least approximate) understanding of how they function and when they break. Approaches for testing such white-box systems exploit this knowledge and typically rely on splitting them into smaller semantic (sub-)units and, hierarchically reversed, first examine these components, their integration, and, finally, the entire system. For non-interpretable (neural) ML models, such as modern medical imaging and autonomous driving systems, such assignments of functional blocks to model parts cannot be made. Thus, traditional testing approaches are not readily applicable to assess learned systems and their uncertainty estimates. Dedicated techniques for testing ML systems, on the

other hand, exist; however, these approaches often do not reflect application-specific safety and real-world demands, such as sufficient data coverage in critical input space regions.

To contribute, in light of the above, to trustworthy ML systems, we require uncertainty estimators

- with well-understood properties and known limitations, which may, for instance, stem from constraints of model capacity;
- that allow for close and stable approximation of (not explicitly) given data distributions;
- and that are assessed by evaluation measures and test strategies that address the (specific) requirements of a given real-world ML use case.

These three research aspects provide the roadmap for this work. In the following, we outline, for each of these aspects, conceptual and technical difficulties and sketch our respective research approaches and contributions.

Impact of model capacity on uncertainty Existing uncertainty mechanisms rely on (implicit) assumptions, for instance w.r.t. model capacity, which are not always guaranteed. Specifically, they often assume simplifying distributional properties, where in a large-capacity limit typically Gaussianity is argued. However, the validity of these assumptions and their limitations are not thoroughly theoretically understood yet and thus bear the risk of unforeseeable behavior and error modes, particularly for capacity-constrained real-world learning systems. Therefore, we analyze the impact of model complexity on the distributional properties of uncertainty estimates as well as on uncertainty quality, i.e., the ability to correctly predict data-intrinsic uncertainty and model insufficiencies. Technically, we control a model's capacity by manipulating its hidden parameter space (see Chapter 4), either explicitly, by varying its size (see Section 4.2), or implicitly, by imposing structures on it that de-facto reduce model expressiveness (see Sections 4.1 and 4.2).

We begin in Section 4.1 with a simple model for uncertain dynamics, namely a hidden Markov model (HMM), and integrate word2vec-inspired (Mikolov et al., 2013b) representation learning into it. In this way, non-trivial embeddings for all model parameters are obtained, and an alternative way to control model complexity is provided. Next, we analyze the quality of the uncertain transitions of this *DenseHMM* as a function of model complexity. The DenseHMM reaches state-of-the-art performance (for HMMs) and thus confirms the effectiveness of our representation learning approach. We identify different model regimes as model expressiveness (steered via representation length) changes.

In the context of neural models, MC dropout is a widely used technique for complexity control that adds uncertainty to otherwise deterministic mappings. Providing, however, only a rough approximation of Bayesian inference, MC dropout also carries theoretical and practical drawbacks (see, e.g., Osband (2016); Gal et al. (2017)). For MC dropout networks, we investigate the different types of output distributions that stochastic neuron

dropout can induce in Section 4.2. The model capacity is controlled via layer widths and correlated initializations of the network parameters. While dropout-evoked output distributions are commonly assumed to exhibit Gaussian behavior, we demonstrate, using an explicit construction, novel properties, namely, *non*-Gaussian asymptotics. This opens up new possibilities to control and model distributional properties of uncertainty.

Modeling uncertainty estimates by means of Wasserstein dropout A better understanding of the "inner mechanics" and the output distributions of probabilistic models, as sketched above, provides a foundation for more reliable uncertainty quantification. To practically employ uncertainty estimators as parts of (open-world) learning systems, their output distributions need to be optimized to (training) data distributions that are typically characterized by ambiguity and sparsity. Therefore, a combination of accurately modeled aleatoric uncertainty and stable generalization to unseen (and potentially structurally different) inputs, i.e., a reliable quantification of epistemic uncertainty, is desirable.

We address these demands with a novel uncertainty mechanism that differently aligns so-called dropout-based sub-networks that are obtained by randomly switching off some of the neurons of the full network. In the context of MC dropout, such sub-networks are used to model NN output distributions in a parameter-free and potentially more stable way, as they allow for encoding uncertainty information in the entire structure of a network. While being widely used for estimating epistemic uncertainty as part of MC dropout, the application of dropout sub-networks to model *input-dependent aleatoric* uncertainty has—to the best of our knowledge—not been considered yet. Taking advantage of the Wasserstein distance (see, e.g., Villani (2008)), we construct a novel uncertainty mechanism, *Wasserstein dropout* (abbreviated as W-dropout, see Chapter 5), and its compute-efficient Gaussian-likelihood one-sample (GL-OS) variant. Both W-dropout and its GL-OS version simultaneously model the (regression) task at hand and input-dependent (also called heteroscedastic) aleatoric uncertainty.

We conduct an extensive empirical evaluation where W-dropout outperforms state-of-the-art uncertainty techniques w.r.t. various benchmark metrics, not only on in-data test sets, but also under (different types of) data shifts. This finding can be understood as W-dropout, which is constructed to grasp heteroscedastic aleatoric uncertainty, moreover, captures epistemic uncertainty—a property "inherited" from the sub-network matching mechanism of MC dropout. Additionally, we adapt W-dropout to modern 2D object detection networks, namely, SqueezeDet (Wu et al., 2017) and RetinaNet (Lin et al., 2017b), and find its before-mentioned beneficial properties confirmed, thus providing evidence that W-dropout is reliably applicable as part of complex, task-specific model architectures.

In our evaluations of W-dropout (and other uncertainty estimators), we focus on model reliability in *individual* scenarios (in contrast to solely dataset-averaged assessments). Moreover, lower "bounds" of system quality are studied, as well as different types of data shifts and potential violations of commonly made modeling assumptions, particularly of Gaussianity. To this end, we introduce two novel uncertainty measures: on the one hand,

a non-saturating calibration score that allows us to determine previously unresolvable types of deviations between a model's predicted uncertainty and its actual error. On the other hand, a measure for distributional tails that facilitates the analysis of data subsets, particularly of worst-case scenarios, w.r.t. uncertainty quality.

Building uncertainty estimators for product-grade deep learning systems Striving for safe and reliable deep learning (DL) systems, it is beneficial to think of uncertainty estimators as "tools" that come along with specific combinations of strengths and weaknesses, e.g., with respect to estimation quality, generalization abilities, and computational complexity. Given a DL application, uncertainty "tools" need to be picked that address its use case-specific (and potentially broad) set of requirements. Recall, for instance, the probabilistic object detection model described above, where uncertainty estimates allow for distinctions between "familiar" and "novel" objects in traffic scenes. This information may serve use cases, such as uncertainty-informed motion planning. In Chapter 6, we structure and shape the requirements of such uncertainty use cases, from both an ML and a safety perspective, and analyze representative uncertainty modeling approaches (and our W-dropout technique) w.r.t. them. We finally develop strategies to systematically test the obtained uncertainty estimates, particularly because established testing approaches for engineered systems are generally not applicable (see above). Instead, our testing strategies build on iteratively refined compilations of test datasets, beginning with coarse data selections to test basic properties of an uncertainty estimator, followed by broad datasets to analyze its global behavior, toward more specific test datasets that represent application-critical (semantic and non-semantic) input space areas to uncover (potential) structural weaknesses of an estimator. Within this data-hierarchical approach, the depth and focus points of testing are "derived" from the requirements of the uncertainty use case. This requirement-based (development and) testing framework is broadly applicable and can help to establish safer ML systems. It also anticipates future machine learning regulations that require evidence for the technical appropriateness of machine learning systems. Our test strategies provide such evidence for system components modeling uncertainty.

Outline In the following, we first provide a brief overview of basic machine learning concepts in Chapter 2, particularly on learning paradigms (Section 2.1), neural networks (Section 2.2), model optimization (Section 2.3), and regularization (Section 2.4). Building on this foundation, we present central aspects of uncertainty quantification in Chapter 3: the main types of uncertainty and their characteristics (Section 3.1), classes of modeling approaches to account for them (Section 3.2), and common scores for the evaluation of uncertainty quality (Section 3.3). The main part of the thesis is structured according to the overview presented above. Studying uncertainty estimation to contribute to trustworthy ML systems, we first investigate the impact of the complexity of (neural) learning systems on the distributional properties and the quality of their uncertainty estimates in Chapter 4. Specifically, we analyze hidden Markov models enhanced by

word2vec-inspired representation learning abilities (Section 4.1) and MC dropout networks (Section 4.2). Targeting the stable approximation of data distributions, we propose a novel dropout-based uncertainty mechanism termed Wasserstein dropout and two complementary uncertainty measures in Chapter 5. To actually harness uncertainty quantification for trustworthy ML applications, we lastly propose a conceptual framework for uncertainty use cases in Chapter 6 that allows for systematically addressing application and regulatory requirements by using uncertainty modeling techniques and hierarchical testing strategies. A summary and a discussion of our main results, as well as an outlook on promising follow-up research activities, conclude the work (see Chapter 7).

2. Basic Concepts

Before addressing the uncertainty of ML model predictions in the following chapters, we first provide an understanding of how these predictions are obtained. To this end, we begin with a concise recapitulation of standard learning paradigms in Section 2.1. Next, we lay out the building blocks of neural networks, the central model class considered in this thesis, in Section 2.2, and review fully connected and convolutional layers. Approaches to optimizing and evaluating (neural) ML models are presented in Section 2.3. Finally, we discuss model regularization in Section 2.4 to provide a basis for subsequent analyses of the impact of model capacity on uncertainty estimation (see Chapter 4).

2.1. Supervised and unsupervised learning

When optimizing models on data, two fundamental learning paradigms can be distinguished: *supervised learning* that seeks to map input features to "target" outputs, and *unsupervised learning* where no such targets are given and the model instead "distills" intrinsic structural dependencies from a dataset.

While supervised learning provides a clear notion of how successful a mapping onto "target" values (also called labels) is, it comes at the cost of requiring pre-defined, task-specific labels. These labels are typically expensive to obtain, as they need to be provided by a human expert. This is in contrast to unsupervised learning, which does not require such labels. A sub-type of unsupervised learning is self-supervised learning that can (in some sense) be understood as supervised learning with labels that are a natural part of the data and thus come "free of charge", for instance, the "future" observations in a historical time series or the tokens of a text. Combinations of self-supervised and supervised learning recently gained renewed attention: self-supervised pre-training (of neural models) on large-scale datasets and a subsequent supervised "fine-tuning" enable powerful language and vision models while relying on only comparably small sets of labeled data (see, e.g., Devlin et al. (2019); Chen et al. (2020)). While we study a model for uncertain dynamics that is trained in an unsupervised fashion in Section 4.1, the focus of this thesis is on supervised learning.

Within supervised learning, most modeling tasks fall into the categories of *classification* or *regression*. These two types of tasks differ by the structure of their output spaces: classification assumes a discrete label space (think, for instance, of the assignments of pictures to categories like "human", "animal" or "countryside" that can be found under the term "automatic tagging" on many modern mobile phones), while for regression tasks the outputs stem from a continuous space (think, e.g., of a navigation system that predicts the

expected time of arrival at a destination). The parts of this thesis concerning supervised learning put an emphasis on regression tasks.

To model a mapping from input features (e.g., the images to be tagged) to labels, we first choose a model class, i.e., we predetermine the type of models to be considered. Model classes differ in their complexity, i.e., their ability to grasp different types of statistical dependencies between inputs and outputs (a linear model, for instance, is unsuited for modeling a strongly nonlinear dependence), and their inductive bias (also called learning bias), i.e., the (often only implicitly given) set of assumptions that a learning algorithm makes when predicting labels for previously unseen input data points. Given a model class, one seeks to find a model within that class which closely approximates the input-output dependencies of so-called *training data*. This is done via optimization, i.e., the parameters of a model are adapted such that the model outputs (more) closely match the desired "target" values. While analytical optimization schemes exist for some model classes, many more complex ones require (often iterative) numerical procedures (see Section 2.3). This process of adjusting the model parameters is called *model training*. It seeks to (iteratively) reduce the deviations between current model outputs and the labels of the training data, as measured by a *loss function*.[1]

The quality of a model for a previously unseen data point, however, cannot be sufficiently evaluated by means of training losses, as the training dataset is commonly assumed to be only one of many possible samples from an unknown (and inaccessible) ground truth (gt) data distribution P_{gt}. Due to the finite size of the training sample, it may, on the one hand, not comprise some relevant statistical patterns of P_{gt} and, on the other hand, may contain idiosyncrasies that a model "internalizes" during optimization. Thus, once the model is optimized, evaluations on another sample from the ground truth distribution, which is called *test dataset*, are conducted. Being drawn from the same data distribution, it is structurally highly similar to the training data, however, contains input data previously unseen by the model. Applying the trained model to such "new" data points is called *inference*. For the reasons outlined above, the resulting performance of the model on these data points is typically below its performance on the training dataset it was optimized on. This difference in quality is termed *generalization gap*. For a discussion on how to steer (or, at least, influence) model complexity and the generalization gap, see Section 2.4 on model regularization.

For some model classes, statistical learning theory provides upper bounds for generalization errors (see, e.g., Vapnik (1991)). Given the *empirical risk* (i.e., the training error), the size of the training dataset and the complexity of a model class as characterized by the Vapnik-Chervonenkis (VC) dimension, the *true risk* (i.e., the model error on the unknown ground truth data distribution) can be bounded. The bounds increase with model complexity and decrease as the size of the training dataset grows. Such theoretical statements typically assume *independent and identically distributed* (i.i.d.) datasets and hold for a broad range of data distributions P_{gt}. In practice, it is, however, often challenging to

[1] It is common to split the "original" training dataset into two parts: the "actual" training set used to minimize the loss function and a (typically smaller) validation set that is employed to determine hyperparameters of the optimization procedure such as its termination "time".

readily apply such results from statistical learning theory. On the one hand, it is difficult to determine the VC dimension of complex models like application-scale neural networks. Moreover, the bounds provided by statistical learning theory are coarse for many complex learning systems and thus of reduced practical significance. For many open-world machine learning (ML) systems, on the other hand, input data points furthermore fail to meet the i.i.d. assumptions. ML-based systems for automated driving, for instance, process sensor streams where subsequent recordings are highly correlated. Operating in the open world, they additionally encounter input data that structurally deviates from training data due to the sheer size of the input space and due to continuous changes of the environment over time.

2.2. Neural networks

Among the various model classes considered in machine learning, *artificial neural networks* are arguably the most discussed one in recent years. They are at the heart of the notable progress achieved in various fields of ML ranging from text and speech understanding (Vaswani et al., 2017; Devlin et al., 2019) over computer vision (He et al., 2016; Chen et al., 2020) to ML for the sciences (Segler et al., 2018; Jumper et al., 2021), to name just a few. In the following, we present the basic building blocks of this model class.

The smallest unit of a neural network is called *neuron*. In its simplest form, it computes a weighted sum of its inputs and applies a (non-)linear transformation to the result. One of the first models built that way was the so-called perceptron, a linear classifier (Rosenblatt, 1958). More complex neural networks employ a multitude of connected neurons that are (in many cases) grouped in so-called *layers*, which in turn are ordered: the first layer, termed *input layer*, takes the components of a given input data point and processes them as described above. The outputs of the input layer serve as input for the first *hidden layer* whose outputs are then handed over to the second hidden layer and so forth until the final layer, the *output layer*, is reached that generates the output values of the network. Unlike the hidden layers, the output layer is often linear for regression networks. For classification networks, the results of the last layer are typically passed through a softmax function as it maps onto the simplex. This way, the network outputs can be interpreted as class probabilities. Neural networks with several hidden layers are termed *deep neural networks*.

More abstract, neural networks can be represented as graphs where graph nodes correspond to the neurons and graph edges to the inter-neuron connections. Neural networks in which information always flows from input to output then correspond to cycle-free graphs. They are called *feedforward networks*. Recurrent neural networks (RNNs, see, e.g., Rumelhart et al. (1985); Hochreiter & Schmidhuber (1997)), in contrast, build on cycle connections to model sequential dependencies. RNNs are not considered in this thesis. In the following, we focus on two widely used types of feedforward networks that are important for the remainder of this work: fully connected networks (FCNs) and convolutional neural networks (CNNs). In FCNs (Werbos, 1974; Rumelhart et al., 1985),

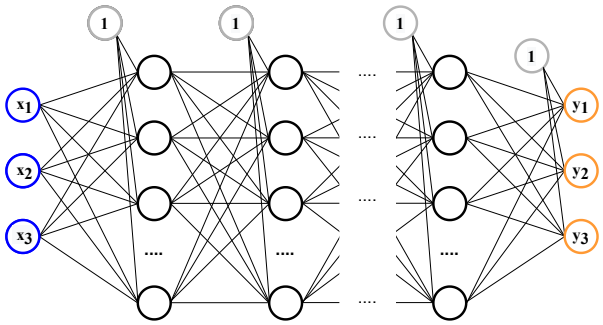


Fig. 2.1.: Sketch of a fully connected neural network with multiple hidden layers that processes three input values (blue circles) and returns three output values (orange circles). The network output is obtained by repeatedly applying (in most cases) nonlinear transformations to the input. This is done within the neurons that are visualized as empty circles. Light gray circles containing a "1" value indicate additive bias terms. For further details, see the text in Section 2.2.

all neurons (also called units) of a hidden layer are connected with all hidden units of the previous and the subsequent layer, while no connections exist within the layer (see Fig. 2.1). Mathematically, a layer of a FCN reads $\phi(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$ with $\boldsymbol{x}$ being the output of the previous layer (or the input to the network), $\boldsymbol{W}$ a matrix of weights, $\boldsymbol{b}$ an additive bias term and ϕ a so-called activation function that is applied element-wise. When optimizing a fully connected network (see Section 2.3), both weights and biases are adapted. The affine transformation $\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ yields the so-called *pre-activations*. Applying the typically nonlinear activation function ϕ, we obtain the *activations* $\boldsymbol{a}(\boldsymbol{x}) := \phi(\boldsymbol{W}\boldsymbol{x}+\boldsymbol{b})$, the outputs of the layer. A fully connected (regression) network $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ is then given by a composition of (in this case, L) fully connected layers as

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{W}^{(L+1)}\, \boldsymbol{a}^{(L)}\Big(\boldsymbol{a}^{(L-1)}\Big(\boldsymbol{a}^{(L-2)}\Big(\ldots\, \boldsymbol{a}^{(1)}(\boldsymbol{x})\, \ldots\Big)\Big)\Big) + \boldsymbol{b}^{(L+1)}\ , \tag{2.1}$$

where $\boldsymbol{\theta}$ summarizes all learnable parameters of the FCN. The network maps inputs from the $\mathbb{R}^{n_{\text{in}}}$ onto outputs from the $\mathbb{R}^{n_{\text{out}}}$. As these types of networks can be seen as extensions of the above described perceptron, they are also called multilayer perceptrons (MLPs).

A central characteristic of a FCN is the repeated application of a nonlinear activation function ϕ, as, without it, Eq. 2.1 reduces to a single affine transformation. Among the various proposed activation functions, some of the most widely used ones are the nonlinear hyperbolic tangent, the nonlinear sigmoid as well as the rectified linear unit (ReLU).

These are defined as

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \ , \tag{2.2}$$

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \ , \tag{2.3}$$

$$\text{ReLU}(x) = \max(0, x) \ . \tag{2.4}$$

In contrast to tanh and sigmoid, ReLU does not saturate for large positive input values, a property that may ease model optimization as so-called vanishing gradients generally occur less often (see Section 2.3 for details). Variants of these activation functions exist that seek to further improve model optimization, for instance, leaky ReLU (Maas et al., 2013), which provides, unlike the standard ReLU, a small learning signal for negative pre-activations. Parametric ReLU (He et al., 2015) extends the idea of leaky ReLU and allows us to learn its slope parameter for negative-valued inputs as part of the model training.

FCNs are widely applied as they allow for approximating a broad of set of functions. A classical result in this regard is due to Hornik et al. (1989) who showed that feedforward networks with one hidden layer are, in the limit of infinite layer width, universal function approximators. Comparable results, moreover, exist for networks with fixed layer width for which the limit of infinitely many hidden layers is considered (see, e.g., Kidger & Lyons (2020)). An extensive theoretical treatment of deep feedforward networks can be found in Roberts et al. (2021).

As the number of learnable parameters of FCNs increases linearly with the width of a hidden layer and the number of input and output components, they do not scale well to high-dimensional data spaces as are given, for instance, for image processing tasks. This observation motivates another type of feedforward architecture, convolutional neural networks (CNNs, Fukushima (1988); LeCun et al. (1999)). Their central building block are *convolutional layers* that exploit the "invariance" of many input features under translation: both low-level features like textures and contours and high-level features such as objects or humans are typically unrelated to their precise position in an image. This "invariance" enables to learn parameters that are decoupled from a specific spatial location. Technically, this is realized by so-called *kernels* (or filters), parameter matrices with a small number of elements that are applied to all patches of an image (or to the output feature maps of the previous convolutional layer). Formally, such a convolutional mapping reads

$$(\boldsymbol{X} * \boldsymbol{K})_{i,j} = \sum_{h=1}^{H} \sum_{w=1}^{W} X_{h+(i-1)s,\, w+(j-1)s}\, K_{h,w} \ , \tag{2.5}$$

where $\boldsymbol{K}$ denotes a kernel matrix of size $H \times W$ and $\boldsymbol{X}$ the input feature map. For a patch of this feature map, element-wise multiplications with kernel elements are performed. Adding a bias term b to the sum of these products and applying an activation function ϕ yields the element (i, j) of the output feature map, $Y_{i,j} = \phi\left((\boldsymbol{X} * \boldsymbol{K})_{i,j} + b\right)$. An

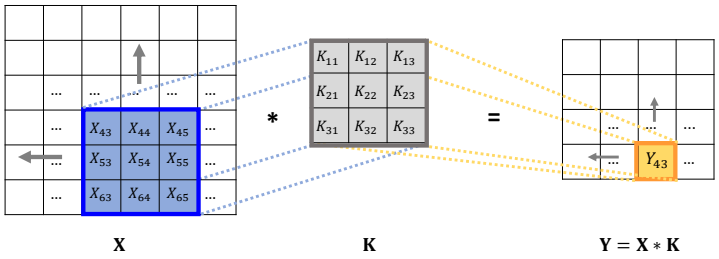


Fig. 2.2.: Schematic illustration of a convolutional mapping. Assuming translational invariance, a learned (in this case 3×3) kernel $\mathbf{K}$ is repeatedly applied to all patches of an input feature map $\mathbf{X}$, yielding an output feature map $\mathbf{Y} = \mathbf{X} * \mathbf{K}$. The elements of $\mathbf{Y}$ are obtained by element-wise multiplication and summation, e.g., $Y_{43} = X_{43}\,K_{11} + X_{44}\,K_{12} + X_{45}\,K_{13} + \ldots$ In this example, the stride of the convolutional mapping is 1, the bias term is 0, and no nonlinear activation function is applied, see the text in Section 2.2 for details. The image is inspired by a visualization from Velickovic (2016).

important technical parameter of a convolutional mapping is the stride s, the "step size" of the kernel's "movement" across the input feature map. This parameter influences the size of the output feature map, i.e., whether and to which degree its spatial "resolution" is reduced relative to the input feature map. For a schematic visualization of such a convolutional mapping, see Fig. 2.2. Early ideas for convolutional mappings such as the Neocognitron were inspired by biological systems (see Fukushima (1988)), as is true for other building blocks of neural networks such as the perceptron (Rosenblatt, 1958). A review of the historical development of neuron-based learning systems can be found in Schmidhuber (2015).

Convolutional neural networks typically extract features from their input by repeatedly applying sets of convolutional filters (and pooling layers, see below). The "receptive field" of the features generated this way, i.e., the patch of the input image they are based on, increases from layer to layer, rendering the extracted features in the first layers low-level and the ones in subsequent layers more high-level. Figuratively, one may think, for the first layers, of simple local features such as contrasts, textures or lines that are iteratively, from layer to layer, "composed" to larger and more complex shapes and structures. Such interpretations, however, hold at best approximately, as the features of CNNs (and NNs in general) are, due to the complexity of the model class, non-interpretable. However, as the parameters of the convolutional filters are learned from data, the resulting features are more flexible compared to classical hand-crafted filters in image processing such as the Sobel-Feldman operator (Sobel & Feldman, 1968) for edge detection, and generalize significantly better. This renders CNNs the de-facto standard for many computer vision tasks.

Within the group of CNNs, we distinguish between two types of networks: those that combine convolutional and fully connected layers and those that are fully convolutional

(see, e.g., Long et al. (2015)), i.e., the latter ones do not require any components from FCNs. Fully convolutional architectures are typically more lightweight compared to the more traditional "combined" architectures that contain a parameter-intense transition between the last convolutional layer and the first fully connected layer. Fully convolutional networks particularly rely on so-called pooling layers (LeCun et al., 1989) to reduce the two "spatial" dimensions of their feature maps. These layers "compress" feature map patches by either reducing them to their mean (average pooling) or by taking their maximum value (max pooling).

While CNNs are traditionally employed for computer vision tasks (such as image classification (Krizhevsky et al., 2012) or object detection (Liu et al., 2016)), they moreover find application in tasks such as time series modeling (see, e.g., Lea et al. (2016)) or drug discovery (see, e.g., Wallach et al. (2015)). Recent computer vision architectures increasingly combine convolutional networks with transformer components, see, e.g., Wu et al. (2020a). In this thesis, we consider, in the context of uncertainty estimation, two object detection architectures, namely, SqueezeDet (Wu et al., 2017) and RetinaNet (Lin et al., 2017b), see Section 5.4.

2.3. Model optimization

To find a model within a model class that closely approximates the input-output dependencies as given in a training dataset, many optimization schemes require a loss function, which is also called objective function, that measures the current deviations between predicted and "target" outcomes. First, we outline characteristics of widely used regression loss functions. Next, we describe gradient-based optimization procedures for neural networks. In the last part of this section, a concise review on parameter estimation from a probabilistic perspective is given.

In regression setups, deviations between target values and model outputs are commonly measured by means of (squared) L2 distances (which are also called Euclidean distances). The average of these squared deviations across a dataset is referred to as the mean squared error (MSE). Alternatively, one may consider absolute model residuals, i.e., the absolute differences (also termed L1 distances) between model outputs and labels, and minimize, for a set of training data points, the corresponding mean absolute error (MAE).

MSE and MAE differ in the uniqueness and the properties of the results obtained when minimizing them. Estimating, for instance, the parameters $\boldsymbol{\theta}$ of a linear model $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{\epsilon}$ with Gaussian noise $\boldsymbol{\epsilon}$, the MSE yields a unique optimal estimator whereas the MAE in general does not (see, e.g., Sielken & Hartley (1973)). In return, the MAE is more robust toward model outcomes largely deviating from the desired "target" values as all model residuals are equally weighted, unlike the MSE that puts (relatively seen) higher emphasis on large deviations.

To leverage the respective strengths of L1 and L2 losses, combinations of them were proposed, e.g., the smooth L1 loss (also called Huber loss, Huber (1992)) that employs an L2 loss for small deviations below a pre-defined threshold value and an L1 loss above

it. This way, higher stability for small deviations as well as lower susceptibility to large deviations is obtained. These characteristics render the smooth L1 loss a standard regression objective in object detection, see, e.g., Fast R-CNN (Girshick, 2015), SSD (Liu et al., 2016) and RetinaNet (Lin et al., 2017b), the last of which is employed in Section 5.4. A generalized regression loss that contains several standard regression losses as special cases (including the L2 and the smooth L1 loss) was proposed by Barron (2019). It depends on two parameters that can be learned as part of the model optimization and that allow for flexible adaption of the robustness of the objective function for each output dimension.

As for many complex model classes (such as DNNs) no closed-form expressions are known for the optimal parameters given a training dataset, numerical optimization schemes are employed that minimize loss functions like the ones outlined above. Moreover, these optimization problems are typically non-convex, i.e., no unique global minimum but many, qualitatively different local minima exist. Numerical routines seek to find one of these local optima by iteratively improving a randomly parameterized initial model. For loss functions that are differentiable w.r.t. the model parameters, this can be done by applying gradient-descent (GD) procedures (Kiefer & Wolfowitz, 1952). For those, an update of the parameters $\boldsymbol{\theta}$ of a model $f_{\boldsymbol{\theta}}$ is given by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \mu_t \nabla \mathcal{L}(\boldsymbol{\theta}_t) = \boldsymbol{\theta}_t - \frac{\mu_t}{N} \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} L\Big(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i\Big) \tag{2.6}$$

with iteration index t, learning rate μ_t and the gradient of the training data loss $\nabla \mathcal{L}(\boldsymbol{\theta}_t)$ for the current model parameters $\boldsymbol{\theta}_t$. The training data loss $\mathcal{L}$ is defined as the arithmetic mean of the loss function values $L\big(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i\big)$ for the training data points $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ with $i = 1, \ldots, N$. As $-\nabla \mathcal{L}(\boldsymbol{\theta}_t)$ points in the direction of the steepest decrease of the training loss, a step in this direction is taken, scaled by the learning rate μ_t. For neural networks, the gradient w.r.t. the model parameters is calculated by back-propagation (Rumelhart et al., 1986), i.e., by the iterative propagation of training losses backward through the network using the chain rule of differentiation.

These gradient descent steps are repeatedly applied until a stopping criterion is fulfilled. Various such criteria exist that range from fixed maximum numbers of iterations to more informed ones that take the progress of the optimization more directly into account, e.g., by stopping when the changes of the training loss fall below a pre-defined (relative) threshold or when overfitting (see Section 2.4) is indicated by an increase of a validation set loss.

Performing each GD step on the entire training data (referred to as full-batch gradient descent) is cumbersome for large datasets and often leads, in particular for neural networks, to local minima in parameter space that do not generalize well. Instead, so-called mini-batch gradient descent (also named stochastic GD (SGD)) is applied where in each optimization step only a small, randomly drawn fraction of the training dataset, termed mini-batch, is used to calculate the loss gradient. These mini-batches are drawn without replacement such that each data point is used once before the sampling procedure starts

all over again. One such cycle is called *training epoch*. The composition and the order of the mini-batches vary from (training) epoch to epoch.

Both theoretical (Amir et al., 2021) and empirical (Keskar et al., 2017; Masters & Luschi, 2018) studies provide evidence for the beneficial generalization properties of SGD-optimized models. The stochasticity in the optimization was shown to ease "escapes" from saddle points (Fang et al., 2019) and to elicit convergence to "broad" (and thus better generalizing) minima (Keskar et al., 2017).

Various extensions of SGD exist, among these, AdaDelta (Adaptive Delta, Zeiler (2012)) and Adam (Adaptive Moment Estimation, Kingma & Ba (2015)) that scale the gradient component-wise, using running averages of gradients and related quantities from previous optimization steps. These scalings can be seen as approximations of the diagonals of inverse Hessians. Thus, AdaDelta and Adam can be understood as rudimentary proxies for second-order optimization methods.

Theoretical results exist that give, under certain assumptions, convergence guarantees for GD and SGD: (full-batch) gradient descent, for instance, was shown to converge to a global minimum, assuming convexity of the optimization objective and a sufficiently small learning rate (see Section 2.1 of Nesterov (2004)). Related results that relax these assumptions can be found in Lee et al. (2016). In the case of smooth objective functions and sufficiently decreasing learning rates, SGD almost surely converges to a local minimum of the empirical risk (see Chapter 5 of Bottou (1999)). An extensive review of optimization techniques in ML can be found in Bottou et al. (2018).

Once optimized, the quality of trained models is evaluated on a previously unseen test dataset (see Section 2.1). For regression models, for instance, by means of the root-mean-square error (RMSE), the square root of the MSE (see above). Performance on test data is, except for toy problems, rarely perfect for reasons such as data noise, data generation processes that do not include driving factors of labels $\boldsymbol{y}$, or due to inadequate modeling choices (see Section 3.1 for a discussion). To acknowledge such insufficiencies more directly, the labels $\boldsymbol{y}$ are often assumed to be probability distributions conditioned on the input variables $\boldsymbol{x}$. Learning a model can then be rephrased as determining the parameters $\boldsymbol{\theta}$ of $P(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{\theta})$ given samples from an unknown conditional ground truth distribution $P_{\mathrm{gt}}(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{\theta}_{\mathrm{gt}})$. In the following, we sketch frequentist and Bayesian procedures to estimate these parameters, namely, maximum likelihood estimation (MLE), maximum a posteriori estimation (MAP) and estimation by means of the full posterior distribution of the model parameters. As the latter distribution is often intractable, variational inference is discussed that approximates probability distributions through optimization.

Maximum likelihood estimation seeks to find parameters $\boldsymbol{\theta}$ that maximize the likelihood of a set of labels $\boldsymbol{Y}_{\mathrm{train}}$ given the input data points $\boldsymbol{X}_{\mathrm{train}}$. Assuming independence between the individual training data points, the likelihood reads

$$\mathrm{L}(\boldsymbol{\theta}) = P(\boldsymbol{Y}_{\mathrm{train}} \,|\, \boldsymbol{X}_{\mathrm{train}}, \boldsymbol{\theta}) = \prod_{i=1}^{N} P(\boldsymbol{y}_i \,|\, \boldsymbol{x}_i, \boldsymbol{\theta}) \ . \tag{2.7}$$

The optimal parameter set $\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}\in\Theta} \mathrm{L}(\boldsymbol{\theta})$ within a parameter space Θ is called maximum likelihood estimate. Inference is then performed by means of $P(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}_{\mathrm{ML}})$.

For practical reasons, especially numerical stability, the likelihood is often not directly maximized. Instead, its negative logarithm is considered.[2] Minimizing this negative log-likelihood (NLL) is, due to the monotonicity of the logarithm, equivalent to maximizing the likelihood. The NLL of the training dataset reads

$$\mathrm{NLL}(\boldsymbol{\theta}) = -\log \mathrm{L}(\boldsymbol{\theta}) = -\sum_{i=1}^{N} \log P(\boldsymbol{y}_i \mid \boldsymbol{x}_i, \boldsymbol{\theta}) \ . \tag{2.8}$$

In the context of regression, a common choice for modeling P are normal distributions as the central limit theorem (see, e.g., Fischer (2011)) ensures the Gaussianity of averages over i.i.d. samples for a broad range of distributions. For a normal distribution in 1D with mean value μ and standard deviation σ, the NLL of a single data point (x_i, y_i) reads

$$\mathrm{NLL}_{\mathrm{normal}}(\mu, \sigma) = \log(\sigma) + \frac{(\mu - y_i)^2}{2\,\sigma^2} + c \tag{2.9}$$

with $c = \log\sqrt{2\pi}$. This Gaussian NLL can be used to illustrate connections between maximum likelihood estimation and empirical risk minimization: let us assume a model that parameterizes the mean of a Gaussian distribution with constant standard deviation,

$$P(y_i \mid x_i, \sigma, \boldsymbol{\theta}) = \mathcal{N}[\mu_i = f_{\boldsymbol{\theta}}(x_i), \sigma] \ . \tag{2.10}$$

The NLL of a training dataset under this model is given as $\sum_i (\log(\sigma) + (\mu_i - y_i)^2/(2\sigma^2) + c)$. Rescaling and shifting this NLL yields $\sum_i (\mu_i - y_i)^2$, the empirical risk when using an L2 loss.

In contrast to the frequentist approach of MLE, Bayesian approaches treat model parameters as random variables. The initial distribution of these random variables, $P(\boldsymbol{\theta})$, which is also called prior distribution, reflects pre-existing knowledge and assumptions concerning the parameters (before observing any data). Bayes' theorem (see, e.g., Section 1.2 of Bishop (2006)) facilitates updates of the prior distribution $P(\boldsymbol{\theta})$ by means of the data likelihood $P(\boldsymbol{Y}_{\mathrm{train}} \mid \boldsymbol{X}_{\mathrm{train}}, \boldsymbol{\theta})$, i.e., the probability of observing labels given the input data. This update procedure yields the posterior parameter distribution

$$P(\boldsymbol{\theta} \mid \boldsymbol{X}_{\mathrm{train}}, \boldsymbol{Y}_{\mathrm{train}}) \propto P(\boldsymbol{Y}_{\mathrm{train}} \mid \boldsymbol{X}_{\mathrm{train}}, \boldsymbol{\theta})\, P(\boldsymbol{\theta}) \ . \tag{2.11}$$

Its normalization factor, the so-called evidence (or marginal likelihood), is independent of the chosen parameters $\boldsymbol{\theta}$ and is thus neglected here.

Bayesian approaches now differ in whether the full posterior distribution or approximations of it are used for inference. Considering only a mode of the posterior distribution

[2] The negative logarithm transforms the often very small (positive) likelihood values into significantly larger (positive) values that are numerically better to handle.

yields a maximum a posteriori estimate (MAP) of the model parameters,

$$\boldsymbol{\theta}_{\mathrm{MAP}} = \arg\max_{\boldsymbol{\theta}\in\Theta} P(\boldsymbol{Y}_{\mathrm{train}} \,|\, \boldsymbol{X}_{\mathrm{train}}, \boldsymbol{\theta})\, P(\boldsymbol{\theta}) \ . \tag{2.12}$$

For inference, we then use the distribution $P(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{\theta}_{\mathrm{MAP}})$ that is structurally similar to the one obtained for MLE (see above). Indeed, MLE can be considered a special case of MAP that results when assuming a uniform prior parameter distribution. Keeping, in contrast to the MAP, the entire posterior parameter distribution, inference is done using the posterior predictive distribution

$$P(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{X}_{\mathrm{train}}, \boldsymbol{Y}_{\mathrm{train}}) = \int_{\Theta} \mathrm{d}\boldsymbol{\theta}\, P(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{\theta})\, P(\boldsymbol{\theta} \,|\, \boldsymbol{X}_{\mathrm{train}}, \boldsymbol{Y}_{\mathrm{train}}) \ , \tag{2.13}$$

i.e., by calculating the expected value of $P(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{\theta})$ under the parameter posterior.

For many model classes the calculation of the exact posterior distribution is intractable, in particular due to its normalization factor, the so-called evidence term, which is an integral (or a sum) over a high-dimensional parameter space. This motivates approximating approaches that can be broadly categorized into Markov chain Monte Carlo (MCMC) and variational inference (VI) techniques. For a detailed comparison of these approximation techniques, see Blei et al. (2017). Speaking loosely, MCMC techniques are theoretically better understood but do not scale well to large model spaces whereas VI approaches exhibit a beneficial scaling behavior at the price of being less well-grounded in theory. Here, we focus on variational inference as it can be used to motivate dropout-based uncertainty techniques (see Section 3.2) that are a central object of study in this work.

In variational inference (VI) approaches, the posterior distribution is approximated by a so-called variational distribution $Q(\boldsymbol{\theta})$ from a pre-specified class of probability distributions. The optimal $Q(\boldsymbol{\theta})$ is determined by minimizing a dissimilarity function between $Q(\boldsymbol{\theta})$ and the true posterior $P(\boldsymbol{\theta} \,|\, \boldsymbol{X}_{\mathrm{train}}, \boldsymbol{Y}_{\mathrm{train}})$. A common choice for this dissimilarity measure is the Kullback-Leibler (KL) divergence $D_{\mathrm{KL}}(\cdot||\cdot)$ (see, e.g., Subsection 1.6.1 of Bishop (2006)). As $D_{\mathrm{KL}}(Q(\boldsymbol{\theta})||P(\boldsymbol{\theta} \,|\, \boldsymbol{X}_{\mathrm{train}}, \boldsymbol{Y}_{\mathrm{train}}))$ still contains the (often intractable) true posterior, it cannot be readily minimized. Instead, it is re-written as

$$D_{\mathrm{KL}}\Big(Q(\boldsymbol{\theta}) \,||\, P(\boldsymbol{\theta} \,|\, \boldsymbol{X}_{\mathrm{train}}, \boldsymbol{Y}_{\mathrm{train}})\Big) = \log P(\boldsymbol{Y}_{\mathrm{train}} \,|\, \boldsymbol{X}_{\mathrm{train}}) - V\Big[Q(\boldsymbol{\theta})\Big] \tag{2.14}$$

where the second term

$$V\Big[Q(\boldsymbol{\theta})\Big] := \mathbb{E}_{Q(\boldsymbol{\theta})}\Big[\log P(\boldsymbol{Y}_{\mathrm{train}} \,|\, \boldsymbol{X}_{\mathrm{train}}, \boldsymbol{\theta})\Big] - D_{\mathrm{KL}}\Big(Q(\boldsymbol{\theta}) \,||\, P(\boldsymbol{\theta})\Big) \tag{2.15}$$

is called *evidence lower bound* (ELBO, see, e.g., Chapter 10 of Bishop (2006)). As $\log P(\boldsymbol{Y}_{\mathrm{train}} \,|\, \boldsymbol{X}_{\mathrm{train}}, \boldsymbol{\theta})$ is independent of $Q(\boldsymbol{\theta})$, minimizing $D_{\mathrm{KL}}(Q(\boldsymbol{\theta})||P(\boldsymbol{\theta} \,|\, \boldsymbol{X}_{\mathrm{train}}, \boldsymbol{Y}_{\mathrm{train}}))$ is equivalent to maximizing the ELBO. This maximization is feasible as the ELBO does not depend on the true posterior distribution $P(\boldsymbol{\theta} \,|\, \boldsymbol{X}_{\mathrm{train}}, \boldsymbol{Y}_{\mathrm{train}})$.

Once optimized, the resulting $Q(\boldsymbol{\theta})$ distribution is used instead of the true parameter posterior for inference. Thus, the predictive posterior distribution is approximated by

$$P(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{X}_{\text{train}}, \boldsymbol{Y}_{\text{train}}) \approx \int_{\Theta} \mathrm{d}\boldsymbol{\theta}\, Q(\boldsymbol{\theta})\, P(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{\theta}) = \mathbb{E}_{Q(\boldsymbol{\theta})}\left[P(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{\theta})\right] \quad . \tag{2.16}$$

Since the integral in Eq. 2.16 can generally not be calculated analytically, it is common to approximate this expectation value over $Q(\boldsymbol{\theta})$ via Monte Carlo (MC) sampling, i.e., by drawing a sample of parameter estimates $\{\boldsymbol{\theta^{(i)}}\}_{i=1}^{N}$ from $Q(\boldsymbol{\theta})$, yielding

$$P(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{X}_{\text{train}}, \boldsymbol{Y}_{\text{train}}) \approx \frac{1}{N} \sum_{i=1}^{N} P\Big(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}^{(i)}\Big) \quad . \tag{2.17}$$

As the comparison and matching of probability distributions is a technical core of uncertainty estimation, we elaborate on (dis-)similarity measures between distributions (also called statistical distances) in the following, namely, on the information-theoretical Kullback-Leibler divergence (see above) and the transport-based Wasserstein distance.

The Kullback-Leibler divergence between two continuous probability distributions P_1 and P_2 reads

$$D_{\text{KL}}(P_1 || P_2) = \int_{\Theta} \mathrm{d}\boldsymbol{\theta}\, P_1(\boldsymbol{\theta})\, \log\left(\frac{P_1(\boldsymbol{\theta})}{P_2(\boldsymbol{\theta})}\right) \quad . \tag{2.18}$$

Taking on positive values for $P_1 \neq P_2$ (and a value of 0 if and only if $P_1 = P_2$), while not being symmetric in its arguments, the Kullback-Leibler divergence can be interpreted as the dissimilarity of a distribution P_2 from a reference distribution P_1. The KL divergence is not a metric due to the asymmetry mentioned above and, moreover, as it does not satisfy the triangle inequality. This is in contrast to the Wasserstein distance (see, e.g., Villani (2008)) that fulfills all axioms of a metric and that originated in the field of optimal transport. Formally, the p-th Wasserstein distance is defined as

$$\text{WS}_p(P_1, P_2) = \left(\inf_{\pi \in \Pi(P_1, P_2)} \int_{\Theta \times \Theta} \mathrm{d}\pi(\boldsymbol{\theta}, \boldsymbol{\theta'})\, d(\boldsymbol{\theta}, \boldsymbol{\theta'})^p\right)^{1/p} \tag{2.19}$$

with probability distributions P_1 and P_2 on Θ, a distance measure d and $p \geq 1$. $\Pi(P_1, P_2)$ denotes the space of joint distributions with marginal distributions P_1 and P_2. The term $\mathrm{d}\pi(\boldsymbol{\theta}, \boldsymbol{\theta'})\, d(\boldsymbol{\theta}, \boldsymbol{\theta'})^p$ can be understood as a cost of moving an infinitesimal element from $\boldsymbol{\theta}$ to $\boldsymbol{\theta'}$. The Wasserstein distance is thus the minimal total cost of "transforming" a distribution P_1 into a distribution P_2 or vice versa. As the transport of piles of soil provides a good intuition of the concept, the Wasserstein distance is also known as the *earth mover's distance*. Its sensitivity to distances between distributions enables, among others, stable learning signals for gradient-based optimization even when strong divergences between model and data are present. This property is exploited, for instance, in Wasserstein generative adversarial networks (Wasserstein GANs, Arjovsky et al. (2017)) that helped

to overcome some of the difficulties connected with the initial KL-divergence-based formulation of GANs (Goodfellow et al., 2014).

2.4. Model regularization

Optimizing models on training data bears the risk of not only learning general statistical patterns in the data but moreover idiosyncrasies, e.g., due to noise or due to the limited size of the training sample. As a consequence, such models often do not generalize well to previously unseen test data and yield, compared to the training error, significantly higher test errors, a phenomenon referred to as *overfitting*. Regularization approaches seek to minimize this generalization gap between training and test data while keeping the training error of the optimized model small. In the context of risk minimization, this goal can be rephrased as aiming for a small true risk.

To better understand how regularization techniques work, we follow Section 2.9 in Hastie et al. (2009) and consider a noisy function $y = f_{\text{gt}}(x) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma)$. Optimizing a model on a training dataset $\mathcal{D}$ sampled from this function, the expected error of the resulting estimator $f_{\mathcal{D}}$ on a test input data point can be decomposed into (squared) bias, variance and irreducible data noise as follows,

$$\mathbb{E}_{\mathcal{D},\epsilon}\left[\Big(y - f_{\mathcal{D}}(x)\Big)^2\right] = \text{Bias}^2_{\mathcal{D}}\Big[f_{\mathcal{D}}(x)\Big] + \text{Var}_{\mathcal{D}}\Big[f_{\mathcal{D}}(x)\Big] + \sigma^2 \ . \tag{2.20}$$

The bias term

$$\text{Bias}_{\mathcal{D}}\Big[f_{\mathcal{D}}(x)\Big] = \mathbb{E}_{\mathcal{D}}\Big[f_{\mathcal{D}}(x)\Big] - f_{\text{gt}}(x) \tag{2.21}$$

describes the deviation between the (deterministic part) of the ground truth label, $f_{\text{gt}}(x)$, and a mean model obtained by averaging over models $f_{\mathcal{D}}$ that result from training on different datasets $\mathcal{D}$ drawn from the data ground truth. The variance term

$$\text{Var}_{\mathcal{D}}\Big[f_{\mathcal{D}}(x)\Big] = \mathbb{E}_{\mathcal{D}}\left[\Big(\mathbb{E}_{\mathcal{D}}[f_{\mathcal{D}}(x)] - f_{\mathcal{D}}(x)\Big)^2\right] \tag{2.22}$$

measures the "spread" between the different trained models. High variances indicate that the outcomes of the trained models strongly depend on the respective training dataset. A trained model with high bias, on the other hand, may indicate insufficient parameter optimization or that the chosen model class is not expressive enough to reflect the given data dependencies. Using instead a more complex model class may decrease model bias, however, often at the price of a higher variance.[3] Regularization techniques are employed to (at least indirectly) steer these trade-off relations between bias and variance.

[3] Recently, evidence was provided for a so-called "interpolation" regime of highly parameterized models where the bias-variance trade-off of the "classical" regime of smaller models does (seemingly) not hold any longer (Nakkiran et al., 2020). This so-called "double descent" hypothesis may help to improve the theoretical understanding of the good generalization abilities observed, e.g., for many neural networks.

On a technical level, various approaches for model regularization exist, among these, adjustments to the learning objective (e.g., by means of penalty terms), changes of the learning algorithm (e.g., SGD and early stopping) and modifications of the model structure during optimization (for a neural network, e.g., by randomly "dropping" neurons). In the following, we motivate and outline some of these techniques.

Structural risk minimization (Vapnik & Chervonenkis, 1974) analyzes model quality and generalization ability by means of the VC dimension, a measure of model complexity. Based on the VC dimension (and the size of the training dataset), probabilistic upper bounds for the true risk can be formulated (for classification models). In practice, however, the VC dimension is difficult to determine for many model classes and easier-to-handle "proxy" measures of model complexity are employed instead, for instance, penalty terms. Common choices for these terms are L1 and L2 norms of the model parameters. L1 penalization typically leads to sparse solutions whereas L2 penalties entail that the absolute values of the model parameters are of comparable size. Weighting a penalty term with a regularization parameter $\lambda \geq 0$ and adding this product to the empirical risk R_{emp} yields the structural risk $R = R_{\text{emp}} + \lambda\, R_{\text{penalty}}$ where the parameter λ determines the relative sizes of the two summands and thus the strength of the regularization. Interestingly, it can be shown that structural risk minimization bears conceptual similarities with MAP estimation, specifically, that L2 regularization can be understood as placing a Gaussian prior on the model parameters. In this case, the regularization strength corresponds to the variance of the prior distribution.

Other types of regularization techniques do not change the optimization objective but modify optimization routines: SGD, for instance, builds on stochastic gradients instead of full-batch gradients for model parameter updates and yields (as discussed in Section 2.3) favorable optimization results compared to GD in many applications. Moreover, one may consider early stopping (Morgan & Bourlard, 1990), i.e., the evaluation of model performance on a dedicated validation dataset after each training epoch (as a proxy for the true risk of the model) where increases of the validation set loss indicate overfitting.

While the previously described regularization techniques are rather model-agnostic, approaches exist that integrate regularizing elements into the structure of a model. Important examples for such techniques in neural networks are batch normalization and dropout-based regularization (which is, for brevity, often just referred to as dropout). Batch normalization (BN, Ioffe & Szegedy (2015)) was introduced to facilitate the optimization of deep neural architectures that traditionally often suffer from vanishing or exploding gradients during backpropagation. On a technical level, BN is a re-scaling of the activations in hidden layers in a way that reduces the distributional differences between training mini-batches. These "standardized" activations were shown to have advantageous effects on the generalization ability of a model (Luo et al., 2019).

Like BN, dropout (Srivastava et al., 2014) extends the network architecture by inserting additional transformations. However, while BN keeps transformed variants of all pre-activations, dropout-based regularization randomly sets the outputs of some neurons to zero in each training step. This is equivalent to "dropping" the respective neurons entirely.

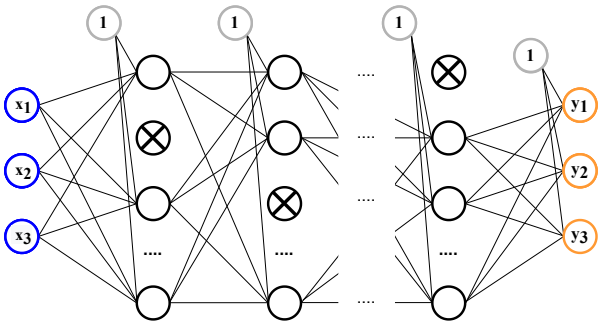


Fig. 2.3.: Schematic visualization of dropout regularization in a fully connected network. In each training step, some neurons of the hidden layers are randomly dropped. This way, so-called co-adaptations of neurons are reduced (see Srivastava et al. (2014)) and, in many cases, better generalization to unseen input data is achieved.

The forward pass is thus performed with a stochastic sub-network instead of the full network (see Fig. 2.3). As a different sub-network is used for each input, optimizing a dropout-regularized network can be understood as the concurrent optimization of various of its sub-networks. Therefore, the network is, figuratively speaking, forced to generate features redundantly and to avoid fragile dependencies between neurons, as each of them may be dropped. Technically, this type of dropout is realized by multiplying each activation in the network with an i.i.d. Bernoulli variable that takes the value 1 with *keep probability* q and the value 0 with *drop probability* $p = 1 - q$. The drop probability p is a hyperparameter that needs to be optimized, e.g., by grid search. As dropout is applied only during training but not at inference, dropout transformations are built in a way that expectation values of activations are left unchanged. Technically, this is done by rescaling the outputs of the dropout transformation with $1/q$ to compensate for the mean value q of the employed Bernoulli variable, i.e., $\boldsymbol{a}_{\text{out}} = 1/q\, \boldsymbol{a}_{\text{in}} \circ \boldsymbol{z}$, with incoming and outgoing activations $\boldsymbol{a}_{\text{in}}$ and $\boldsymbol{a}_{\text{out}}$ and a Bernoulli "mask" $\boldsymbol{z}$.

Multiple variants of dropout exist, some of which are adapted to specific neural architectures like CNNs. Drop-connect (Wan et al., 2013), for instance, drops subsets of model *weights* instead of activations. Spatial dropout (Tompson et al. (2015), also termed channel dropout or 2D dropout) and drop-block (Ghiasi et al., 2018), in contrast, are applied to activations. However, they zero out *groups* of activations (entire feature maps and patches of feature maps, respectively) instead of individual feature map elements as the "reconstruction" of the latter may not be challenging enough for sufficient regularization given spatial correlations in the feature maps.

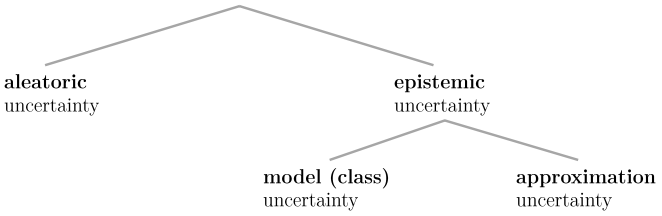
3. Uncertainty Estimation in Machine Learning

Statistical models are inherently uncertain. They are, as outlined in the previous chapter, optimized on training datasets of limited size and varying quality, and their "final" parameter configuration often depends on random initializations and stochastic optimization processes (think, for instance, of the random composition and order of mini-batches employed in SGD update steps, as described in Section 2.3). This raises the question of whether model predictions for previously unseen input data points can be relied on and how their degree of reliability can be quantified. A promising way forward, in this regard, is model-internal self-assessment via uncertainty quantification (see Chapter 1 for a motivation). Here, we lay out the basic technical concepts of uncertainty estimation. In detail, this chapter is organized as follows: In Section 3.1, we provide an overview of the main types of uncertainty present in ML models. Next, in Section 3.2, we review prototypical approaches to model uncertainty and analyze how they capture different types of uncertainty. Evaluating the resulting estimates is non-trivial, as no unique "gold standard" for uncertainty quality is given, mainly due to a lack of uncertainty ground truth labels (see discussion in Chapter 1). Thus, a variety of uncertainty metrics are commonly used. We outline their assumptions, strengths, and weaknesses in Section 3.3.

3.1. Types of uncertainty

Uncertainties are typically categorized by their source. On a high level, distinctions between data-intrinsic uncertainty (also called aleatoric uncertainty) and uncertainty involved in the process of model building (referred to as epistemic uncertainty) are common. At a lower level, however, categorizations of uncertainty types (e.g., Gal (2016); Malinin & Gales (2018); Liu et al. (2019)) differ, for instance, in the considered sub-types of epistemic uncertainty. In this chapter, we loosely follow Hüllermeier & Waegeman (2021), who subdivide epistemic uncertainty into model (class) uncertainty and approximation uncertainty (see Fig. 3.1). In the following, we begin, however, on a higher conceptual level, namely, with the distinction between aleatoric and epistemic uncertainty.

Aleatoric uncertainty, which is also called data(-inherent) uncertainty or data noise, is either a foundational part of the system to be modeled (think, for example, of the stochastic nature of a dice roll) or stems from the process of measuring it, as any sensor (whether it is, e.g., optical, acoustical or tactile) comes with physical limitations. One may, for instance, think of an image that shows an object placed in front of a background. Inevitably, there are pixels at the "border" between foreground and background that



origin	data-inherent	model class limitations	limited information on data ground truth, optimization procedures
example	noisy measurements	linear function for non-linear data	training dataset does not cover all relevant regions of the input space
illustration	Fig. 3.2	Fig. 3.3	Fig. 3.4

Fig. 3.1.: Categorization and characterization of uncertainty sources, following Hüllermeier & Waegeman (2021).

cannot be assigned unambiguously to one or the other. Moreover, not only segment borders but entire segments may not be resolvable (see the medical imaging example in Chapter 1). This aleatoric uncertainty is, for a given system and data generation process, irreducible, i.e., it cannot be eliminated by conducting additional measurements.

Aleatoric uncertainty is further characterized by its input dependence. If the occurrence and strength of the noise vary between input points, it is referred to as *heteroscedastic* whereas the input-independent case is termed *homoscedastic*. For 1D toy examples of both types of noise, see Fig. 3.2. Probabilistic modeling approaches reflect aleatoric uncertainty as they assume conditional distributions $P(y \mid x)$ instead of deterministic functional mappings (see Section 2.3).

In contrast to data-intrinsic uncertainty, *epistemic uncertainty* is due to the process of model building, i.e., due to the chosen model class and optimization routine. Following Hüllermeier & Waegeman (2021), epistemic uncertainty can be regarded as a lack of information on an ideal function

$$f_*(\boldsymbol{x}) := \arg\min_{\hat{\boldsymbol{y}} \in \mathcal{Y}} \int_{\mathcal{Y}} \mathrm{d}y \, P_{\mathrm{gt}}(\boldsymbol{y} \mid \boldsymbol{x}) \, L(\boldsymbol{y}, \hat{\boldsymbol{y}}) \quad \text{for each } \boldsymbol{x} \in \mathcal{X} \ , \tag{3.1}$$

where L denotes a loss function and $\mathcal{X}$ and $\mathcal{Y}$ input and output spaces. By definition, the function f_* minimizes the true risk $\mathrm{R}(f) = \mathbb{E}_{P_{\mathrm{gt}}(\boldsymbol{x},\boldsymbol{y})}\left[L(\boldsymbol{y}, f(\boldsymbol{x}))\right]$ and is thus the best possible point estimator for the supervised learning task given by the conditional ground truth distribution $P_{\mathrm{gt}}(\boldsymbol{y} \mid \boldsymbol{x})$. As this ground truth distribution is typically not accessible, f_* is generally unknown. More information on f_*, however, can be acquired by drawing larger data samples from $P_{\mathrm{gt}}(\boldsymbol{y} \mid \boldsymbol{x})$. In this sense, epistemic uncertainty is reducible. For practical modeling, it is beneficial to consider two sub-concepts of epistemic uncertainty, namely, *model (class) uncertainty* and *approximation uncertainty*.

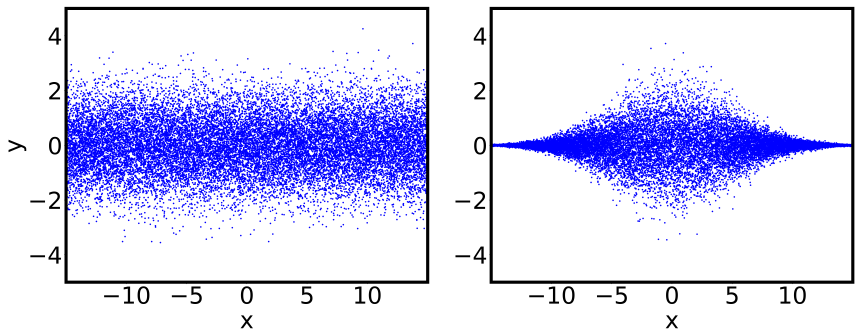


Fig. 3.2.: Two 1D toy datasets with aleatoric uncertainty that is input-independent (l.h.s.) and input-dependent (r.h.s.), respectively. Both data distributions are Gaussian, $x \to \mathcal{N}(0, \sigma(x))$, with $\sigma(x) = \sigma = 1$ on the left-hand side and $\sigma(x) \propto e^{-x^2}$ on the right-hand side.

Model (class) uncertainty refers to the risk that the perfect model f_* may not be part of the considered model class. Choosing, for instance, linear functions for a complex, nonlinear dataset, the solution space will not contain an appropriately fitting model. Another example is visualized in Fig. 3.3 where a small two-layer FCN is fitted to a high-frequency ground truth dataset (blue curve). The (mean) outputs of the trained model (shown in orange) do not reflect the fluctuations of the training data and instead roughly approximate the mean values of its oscillations. This behavior indicates insufficient model complexity for the given dataset and deeper or wider neural networks may be considered to reduce this model (class) uncertainty.

Approximation uncertainty, on the other hand, refers to the discrepancy between i) the (unknown) best model for a learning task within the chosen model class and ii) the model that results from optimization on training data. This type of uncertainty is mainly driven by a lack of knowledge about the ground truth data distribution. The larger the structural deviations between training data and ground truth distribution, the larger the mismatches between the true risk $R(f)$ one seeks to minimize (see Eq. 3.1) and the empirical risk $R_{\text{emp}}(f)$ that is actually minimized. Approximation uncertainty may thus be reduced by improving the data generation process, targeting higher representativeness of the training data. In case that such an (approximately) "bias-free" data sampling can be established, the gap between empirical and true risk can be further reduced by increasing the size of the training set (see discussion in Section 2.1).

Approximation uncertainty manifests in trained models as *parameter uncertainty* that in turn induces output uncertainty. The latter type of uncertainty is visualized in Fig. 3.4 for a 1D toy dataset: the data sample (blue points) drawn from a sinusoidal ground truth curve (dashed blue curve) is not uniformly distributed along x but focuses on the antinodes of the sinus curve, barely sampling the slopes between them. Consequently, a model trained on this dataset (red curve) carries larger output uncertainty for these sparsely covered regions of the input data space (gray intervals surrounding the red curve).

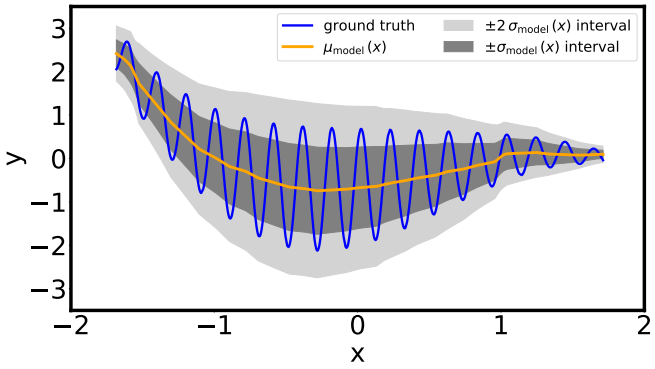


Fig. 3.3.: Illustrative visualization of model class uncertainty. The complexity of the trained neural network (orange) is not sufficient to closely approximate the high-frequency ground truth dataset (blue). Instead, the neural network roughly interpolates the mean values of the data oscillations. In this situation, the benefits of probabilistic models (see Section 3.2) are apparent as they, at least, provide coarse, fluctuation-averaged uncertainty estimates that reflect the magnitudes of the model residuals. The intervals $\mu_{\text{model}}(x) \pm \sigma_{\text{model}}(x)$ and $\mu_{\text{model}}(x) \pm 2\,\sigma_{\text{model}}(x)$ provided by a parametric modeling approach (see ibid.) are shown in dark gray and light gray, respectively.

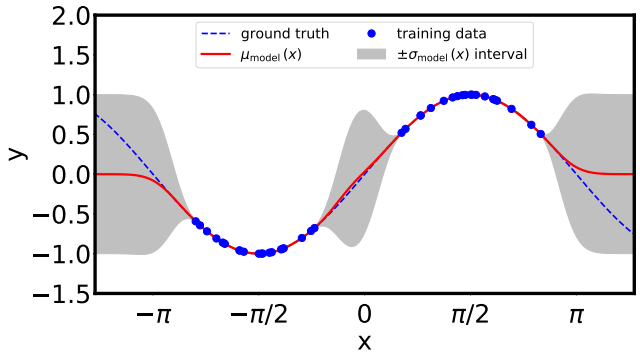


Fig. 3.4.: Schematic visualization of predictive uncertainty that is induced by model parameter uncertainty. The latter one, in turn, is due to a lack of knowledge about the ground truth data. This lack of knowledge arises as the sinusoidal ground truth (dashed blue curve) is sampled in a non-uniform way (along the x-axis), resulting in a dataset (blue points) that barely covers the slopes between the antinodes. A probabilistic model optimized on this data sample is thus subject to epistemic uncertainty, which is low near the training data points and grows with the distance to them (see the gray bands around the red curve that indicates the model's mean predictions). To create this image, the publicly available source code accompanying Wang (2020) was used.

3.2. Modeling uncertainty in neural networks

While all models are subject to (some of) the types of uncertainty described above, uncertainty is often not accounted for by standard modeling techniques. Uncertainty estimation seeks to enhance (neural) models by according capabilities and attempts to (at least approximately) quantify their lack of knowledge for a given input. Precise uncertainty quantification, however, is challenging as various sources of uncertainty need to be grasped while dealing with often high-dimensional input and output spaces and the absence of uncertainty labels. To anyhow facilitate uncertainty quantification in neural networks, most approaches (implicitly or explicitly) make distributional assumptions, especially on the Gaussianity of model outputs. Normality in this regard can be motivated via the central limit theorem when assuming that modeling involves averaging across various i.i.d.-distributed "micro-sources" of uncertainty which, for their part, cannot be modeled explicitly. On a technical level, approaches to estimate predictive uncertainties can be broadly categorized into three groups: parametric models, Bayesian approximations and frequentist techniques. Representative approaches from these three groups are sketched in the following paragraphs.

Parametric approaches Parametric uncertainty methods assume that the modeled output distributions lie within a pre-specified family of probability distributions. This rather strict assumption is technically implemented by networks that map inputs onto parameter values of the considered probability distributions. In the case of a 1D Gaussian output distribution, for example, the network outputs are interpreted as mean $\mu_{\boldsymbol{\theta}}(\boldsymbol{x})$ and standard deviation $\sigma_{\boldsymbol{\theta}}(\boldsymbol{x})$ (see Nix & Weigend (1994); Heskes (1996)). The optimization of these networks is done by minimizing the negative log-likelihood (NLL) of the respectively assumed output distribution (see the discussion on the MLE in Section 2.3). In the 1D Gaussian case, the positivity of the value of the output neuron corresponding to the standard deviation σ is typically ensured by applying a transformation from $\mathbb{R}$ to $\mathbb{R}^+$, such as the softplus$(y) = \log(1 + e^y)$. Generalizing to n-dimensional Gaussian output distributions, the positive semi-definiteness of covariance matrices $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x})$ can, for instance, be guaranteed by constructing $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x})$ from a triangular matrix $\boldsymbol{L}_{\boldsymbol{\theta}}(\boldsymbol{x})$ via $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{L}_{\boldsymbol{\theta}}(\boldsymbol{x})\boldsymbol{L}_{\boldsymbol{\theta}}^T(\boldsymbol{x})$, where the elements of $\boldsymbol{L}_{\boldsymbol{\theta}}(\boldsymbol{x})$ are outputs of the neural network (Harakeh et al. (2020), for instance, use such a construction in the context of object detection).

A technique that combines parametric output distributions with Bayesian elements is deep evidential regression (Amini et al., 2020). It treats the output parameters μ and σ of Gaussian likelihood distributions as random variables that follow a normal-inverse gamma (NIG) distribution. The specific choice of a NIG is made for analytical tractability, as the NIG is the conjugate prior for the assumed observation likelihood, namely, for a normal distribution with unknown mean and variance. This way, a closed-form expression for the

model evidence can be derived,

$$P(y \,|\, \boldsymbol{\eta}_{\boldsymbol{\theta}}(\boldsymbol{x})) = \int_{\Xi} \mathrm{d}\boldsymbol{\xi}(\boldsymbol{x})\, P(y \,|\, \boldsymbol{\xi}(\boldsymbol{x}))\, P(\boldsymbol{\xi}(\boldsymbol{x}) \,|\, \boldsymbol{\eta}_{\boldsymbol{\theta}}(\boldsymbol{x})) = \mathrm{St}\left(y \,\middle|\, \gamma, \beta\frac{1+\nu}{\nu\,\alpha}, 2\,\alpha\right) \quad , \tag{3.2}$$

where St denotes the Student's t-distribution (see Subsection 2.3.7 of Bishop (2006)), $\boldsymbol{\eta} = (\alpha, \beta, \gamma, \nu)$ the four parameters of the NIG distribution that are modeled as network outputs and $\boldsymbol{\xi} = (\mu, \sigma)$ the Gaussian parameters integrated over the domain $\Xi = \mathbb{R} \times \mathbb{R}^+$. The corresponding NN is trained by optimizing the negative logarithm of the model evidence (together with an additive regularization term) by means of gradient-descent procedures. A prediction of the trained model, $\mathbb{E}[\mu] = \gamma$, then comes along with aleatoric uncertainty, $\mathbb{E}[\sigma^2] = \beta/(\alpha - 1)$, and epistemic uncertainty, $\mathrm{Var}[\mu] = \beta/(\nu(\alpha - 1))$.

While this thesis is concerned with the narrow definition of parametric approaches given above, one can define a broader notion. For this, the constraint that network outputs correspond to parameters of probability distributions is relaxed. Quantile regression techniques like Chung et al. (2021b), for instance, are distribution-free and predict multiple quantiles of the input-conditioned target distributions (instead of mean and standard deviation parameters). Moreover, one may even abstract from directly modeling statistical properties of an output distribution, and, instead, construct uncertainty estimates from learned scales in latent spaces, as is done, e.g., by van Amersfoort et al. (2020). Corresponding approaches are often summarized under the term *deterministic uncertainty methods*, see Postels et al. (2022) for a review.

However, the deterministic uncertainty estimates obtained by the outlined techniques are often ill-calibrated, i.e., they do not adequately reflect the actual model quality. This is commonly addressed by means of additional deterministic mappings termed (post-)calibrators that are optimized on holdout validation datasets. On a technical level, a variety of approaches to calibration were put forward, see, e.g., Guo et al. (2017), Kuleshov et al. (2018), Kumar et al. (2019), and Gupta et al. (2021). However, concerns were raised regarding their stability under data shifts, compare, e.g., Snoek et al. (2019), and thus their usability in real-world applications.

Bayesian approaches Bayesian approaches for uncertainty quantification treat model parameters as random variables (see Section 2.3). Assuming a prior parameter distribution $P(\boldsymbol{\theta})$ that reflects pre-existing beliefs and a likelihood $P(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{\theta})$ for labels $\boldsymbol{y}$ conditioned on the model parameters $\boldsymbol{\theta}$, Bayes' rule can be applied to determine the parameter posterior distribution $P(\boldsymbol{\theta} \,|\, \boldsymbol{X}_{\text{train}}, \boldsymbol{Y}_{\text{train}})$ that describes the parameter uncertainty of the trained model. Inference is then performed by means of the posterior predictive distribution, $P(\boldsymbol{y} \,|\, \boldsymbol{x}, \boldsymbol{X}_{\text{train}}, \boldsymbol{Y}_{\text{train}})$, which is given as the expectation value of the likelihood under the parameter posterior (see Section 2.3 for details).

While prior distribution and likelihood are chosen to closely approximate the assumed statistical dependencies of the system to be modeled, in practice, their choice moreover underlies the constraint of enabling efficient (at least approximate) calculation of parameter posterior and posterior predictive distribution, e.g., by employing the so-called conjugate

prior distribution for a likelihood to ensure that both the prior and the posterior parameter distribution lie within the same family of probability distributions.

When neural networks are employed to model these probability distributions, they are termed *Bayesian neural networks* (BNNs). Due to the strongly nonlinear nature of most NNs, the exact calculation of the posterior and the predictive distribution is often infeasible and approximations are required. Different technical approaches to BNNs can be categorized by the types of approximations they apply. In the following, we consider Laplace approximations as well as VI- and MCMC-based techniques.

The Laplace approximation (see Section 4.4 of Bishop (2006)) seeks to simplify the parameter posterior distribution by approximating it with a Gaussian distribution around the maximum a-posteriori (MAP) estimate $\boldsymbol{\theta}_{\text{MAP}}$. Practically, the MAP can be obtained by optimizing a regression model with an L2-regularized L2 objective (see Section 2.3). The Laplace approximation of the parameter posterior reads

$$P(\boldsymbol{\theta} \mid \boldsymbol{X}_{\text{train}}, \boldsymbol{Y}_{\text{train}}) \approx \mathcal{N}\Big[\boldsymbol{\theta}_{\text{MAP}}, \boldsymbol{H}^{-1}(\boldsymbol{\theta}_{\text{MAP}})\Big](\boldsymbol{\theta}) \ , \tag{3.3}$$

where $(\boldsymbol{H})_{ij} = -\partial^2 / \partial\theta_i \, \partial\theta_j P(\boldsymbol{\theta} \mid \boldsymbol{X}_{\text{train}}, \boldsymbol{Y}_{\text{train}})$ denotes the negative Hessian of the parameter posterior. As the calculation and inversion of Hessians is prohibitively expensive for large models, various approximation techniques were put forward: simple approximations, for instance, keep only the diagonal terms of $\boldsymbol{H}$ and neglect all non-diagonal ones (Becker & LeCun, 1989). More advanced approaches like in Martens & Grosse (2015); George et al. (2018); Ritter et al. (2018) exploit that neural models are often structured by layers, which typically leads to Hessians that exhibit a block structure due to differences between intra- and inter-layer dependencies. Such block structures are then modeled using, e.g., Kronecker factorizations (Loan, 2000).

The Laplace-approximated parameter posterior (see Eq. 3.3) is then used, instead of the intractable exact parameter posterior, to calculate the posterior predictive distribution, which is used for inference. This calculation typically involves a high-dimensional integral over the parameter space that is, in most cases, numerically approximated by MC sampling (see Section 2.3).

Another class of approaches to BNNs is based on variational inference (see ibid.), i.e., the parameter posterior distribution is approximated by a variational distribution that typically belongs to a family of probability distributions parameterized by $\boldsymbol{\theta}$. The best approximating variational distribution within this family is determined by maximizing the ELBO w.r.t. $\boldsymbol{\theta}$ as this optimization is equivalent to minimizing the KL divergence between variational distribution and exact parameter posterior. To ease the calculation and optimization of the ELBO, it is common to make simplifying assumptions on the structure of the variational distribution such as Gaussianity and independence between the variational parameters, which allows, for instance, for factorization. While the exact optimization of the ELBO may be possible for small networks (see, e.g., Hinton & van Camp (1993)), further approximations are required for larger models. Bayes by backprop (Blundell et al., 2015), for example, applies MC sampling to estimate the ELBO and

enables its gradient-based optimization by means of a so-called reparameterization trick (Opper & Archambeau, 2009; Kingma & Welling, 2014) that allows, loosely speaking, for back-propagating through random variables. At inference, the posterior predictive distribution is, as for Laplace-approximation-based BNNs, approximated by means of MC sampling.

For some neural uncertainty techniques, connections to the well-understood framework of Gaussian processes (GPs, Rasmussen (2003)) can be established, which help to strengthen the theoretical foundation of the former ones. Before we describe according connections in the next paragraph, we first present basic properties of GPs in the following: Gaussian processes are generalizations of multivariate Gaussian distributions that define probability distributions in infinite-dimensional function spaces. A prior distribution in such a function space is determined by a mean function $m(x)$ that is often assumed to be zero and a covariance function $k(x, x')$, where x and x' are any two inputs. The covariance function pre-determines central properties of the considered functions like their smoothness, i.e., the length scale on which these functions "change". Observing training data points, the prior distribution is updated and functions that do not fit the data well are no longer considered for inference. To make predictions for previously unseen inputs, the defining property of GPs is exploited, namely, that any finite number of observations follows a Gaussian distribution. Starting from the joint distribution of train and test data and conditioning it on the train data, the probabilistic test outputs can be determined. As this calculation involves the inversion of the covariance matrix $k(\boldsymbol{X}_{\text{train}}, \boldsymbol{X}_{\text{train}})$ on training data $\boldsymbol{X}_{\text{train}}$, standard GPs do not scale well to large datasets. Various approximations were put forward to improve their scalability, for instance, stochastic variational GPs (SVGP, Hensman et al. (2013)) and sparse GP regression (SGPR, Titsias (2009)) that apply approximate VI techniques. Probabilistic quality guarantees for these VI approximations are provided by Burt et al. (2019). An extension of GPs that is important for Section 4.2 of this work are so-called deep GPs (Damianou & Lawrence, 2013) that can, grossly simplified, be understood as "concatenations" of GPs where the outputs of the "intermediate" GPs are interpreted as latent variables that serve as input for the subsequent GP. Connections between deep GPs and MLPs can be established by means of neural uncertainty estimation techniques like *Monte Carlo (MC) dropout* that is discussed in the following.

Monte Carlo dropout (Gal & Ghahramani, 2016a) is closely related to the dropout regularization technique (Srivastava et al., 2014) that was outlined in Section 2.4 as both techniques add stochasticity to an otherwise deterministic network by randomly dropping a fraction of its neurons for each input during optimization. However, while the optimization procedures of dropout-regularized and MC dropout networks are alike, their are operated in different ways during inference: for dropout-regularized networks, neuron dropping is switched off, whereas it is kept active for MC dropout. Performing multiple forward passes with a MC dropout network for a given test input yields a set of outputs that is interpreted as a sample of an output distribution.

A Bayesian motivation of MC dropout can be provided by showing an equivalence between certain deep Gaussian processes and dropout MLPs, which allows for interpreting

MC dropout as a variational inference technique. The according proof by Gal & Ghahramani (2016a) starts from a deep GP with a specifically chosen covariance function. As the posterior distribution of this GP is intractable, it is approximated by a Bernoulli-based variational distribution. Approximating the KL divergence between this variational distribution and the GP posterior by Monte Carlo integration is then shown to yield an L2-regularized loss of a dropout MLP. Principled adaptations of Monte Carlo dropout for convolutional (Gal & Ghahramani, 2015) and recurrent (Gal & Ghahramani, 2016b) neural architectures exist.

The widespread use of MC dropout for uncertainty estimation can be attributed to a combination of advantages that it comes along with: given an existing neural model, MC dropout does not require to modify its objective function and necessitates only minor changes to its architecture, namely, the insertion of so-called dropout layers that "mask" a subset of the model's neurons. Standard deep learning frameworks like tensorflow (Abadi et al., 2016) and pytorch (Paszke et al., 2019) provide implementations of such dropout layers as dropout-based regularization techniques, which build on the same types of layers, are widely used. With dropout layers being available "off-the-shelf", MC dropout can be readily implemented. It, moreover, enables uncertainty estimation in dropout-regularized models (like Zagoruyko & Komodakis (2016); Huang et al. (2017); Chollet (2017)) without requiring re-training and scales to application-size neural networks unlike many other Bayesian techniques.

However, these advantages of MC dropout are partially offset by the computational overhead that results from performing multiple forward passes and, moreover, by limitations of its uncertainty modeling capabilities. While its ability to capture epistemic uncertainty is a hallmark of MC dropout, it is less suited to accurately model aleatoric uncertainty (see Section 5.1 for a discussion). To improve on these shortcomings, several extensions of MC dropout were put forward:

Kendall & Gal (2017), for instance, combine MC dropout with networks that parameterize Gaussian distributions with their outputs. This way, MC dropout's ability to model aleatoric uncertainty can be improved. The *mean* prediction of the resulting "Gaussian" dropout model is then given by averaging over the MC dropout-induced distribution of the network's μ-outputs. The *variance* of the model predictions is composed of two terms, on the one hand, the mean value of the σ-output of the network (which describes data-intrinsic uncertainty) and, on the other hand, the MC dropout-induced variance of the μ-output values (that models parameter uncertainty). For a dropout-based uncertainty mechanism that captures heteroscedastic aleatoric uncertainty in a *fully non-parametric* way, see Chapter 5.

Concrete dropout (Gal et al., 2017) improves the flexibility of MC dropout by learning layer-specific drop probabilities as part of the model optimization. To do so, Concrete distributions (Maddison et al., 2017) are employed, continuous relaxations of discrete variables that render the employed learning objectives, in contrast to MC dropout, differentiable w.r.t. the drop rates. Optimization of Concrete dropout networks is done by means of VI, namely, by maximization of an ELBO.

To avoid multiple stochastic forward passes during inference with MC dropout networks, Postels et al. (2019) propose to approximate dropout-based (posterior) sampling by analytical covariance propagation (see, e.g., Taylor (1996)). Another approach to cut down computational overhead is to apply dropout only to the last layers of a network (see, e.g., Snoek et al. (2019)). This way, the propagation of an input through the first layers of the network is deterministic, rendering multiple repetitions of this part of the forward pass unnecessary. The resulting hidden activations then serve as input for all stochastic propagations through the subsequent dropout layers of the network.

Another way to realize BNNs is by means of Markov chain Monte Carlo (MCMC) methods that allow for sampling probability distributions (see Section 11.2 of Bishop (2006)). These sampling techniques are especially useful for multi-dimensional probability distributions when direct sampling is infeasible. An important MCMC technique is the Metropolis-Hastings algorithm (Hastings, 1970) that contains the initial MCMC algorithm (Metropolis et al., 1953) as a special case. The Metropolis-Hastings algorithm allows for sampling from distributions $P(\boldsymbol{\theta})$ that are otherwise difficult to assess. To do so, a function $f(\boldsymbol{\theta})$ is required that is proportional to the density of $P(\boldsymbol{\theta})$. Starting from a random initial point $\boldsymbol{\theta}^{(0)}$ in parameter space, further points $\boldsymbol{\theta}^{(t)}, t \in \mathbb{N}$, are iteratively generated by means of a Markov chain (see Subsection 11.2.1 of Bishop (2006)). Being at point $\boldsymbol{\theta}^{(t)}$ in iteration t, a so-called candidate point $\boldsymbol{\theta}'$ is drawn by means of a proposal distribution $g(\boldsymbol{\theta}' \,|\, \boldsymbol{\theta}^{(t)})$. A simple mechanism, based on the ratio $f(\boldsymbol{\theta}')/f(\boldsymbol{\theta}^{(t)}) = P(\boldsymbol{\theta}')/P(\boldsymbol{\theta}^{(t)})$, is then used to decide whether the candidate point is accepted, i.e., $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}'$, or rejected. It can be shown that the sets $\{\boldsymbol{\theta}^{(t)}\}_{t=1}^{T}$ generated along this chain are samples of the distribution $P(\boldsymbol{\theta})$ if T is large enough to reach the stationary state of the Markov process. Due to the sequential nature of data generation along the Markov chain, correlations between the sample points $\boldsymbol{\theta}^{(t)}$ likely occur.

Such MCMC techniques are beneficial for Bayesian inference as they facilitate sampling of the posterior parameter distribution without requiring to determine its often intractable normalization factor. Instead, the proportionality between posterior distribution and the product of prior distribution and likelihood (see Section 2.3) can be exploited. The MCMC-based samples from the posterior then allow for approximating the posterior predictive distribution of the model.

A variant of the classical Metropolis-Hastings algorithm outlined above is Hamiltonian Monte Carlo (Neal, 2010) that helps to generate candidate points $\boldsymbol{\theta}'$ at larger distances from $\boldsymbol{\theta}^{(t)}$ while maintaining high acceptance rates. This way, correlations between sample points are reduced, which facilitates more efficient sampling of a distribution $P(\boldsymbol{\theta})$ compared to the (often Gaussian) random walks of classical techniques. However, as Hamiltonian Monte Carlo requires gradient computations on the entire dataset in each optimization step, it does not scale well to larger problems. To overcome these limitations, stochastic gradient MCMC techniques were proposed, e.g., stochastic gradient Langevin dynamics (Welling & Teh, 2011) and stochastic gradient HMC (Chen et al., 2014).

Mandt et al. (2017) show that the SGD-based optimization of a model can be interpreted as MCMC sampling from its posterior parameter distribution. In particular, they propose

to approximate the posterior by iterate average stochastic gradient sampling (IASG) that is based on averages over segments of the model's parameter space trajectory during optimization. A conceptually similar approach is the stochastic weight averaging Gaussian technique (SWAG, Maddox et al. (2019)) that approximates a model's posterior distribution by sampling from $\mathcal{N}[\boldsymbol{\mu}_{\mathrm{SWAG}}, \boldsymbol{\Sigma}_{\mathrm{SWAG}}](\boldsymbol{\theta})$ where the mean $\boldsymbol{\mu}_{\mathrm{SWAG}}$ and the covariance $\boldsymbol{\Sigma}_{\mathrm{SWAG}}$ are constructed from the SGD trajectory. To better capture the multi-modal structure of neural network weight posteriors, Zhang et al. (2020b) propose cyclical MCMC that explores and samples from multiple, structurally different local minima (modes of the posterior) during optimization.

Frequentist approaches Unlike Bayesian approaches, frequentist techniques treat model weights as deterministic parameters and not as random variables. Probabilistic (data) dependencies are reflected by (implicit) conditional distributions $P(y \,|\, x)$ that are typically approximated by sampling. A common approach to obtain such samples are *model ensembles*, i.e., sets of models where each model provides a prediction for a given input. The spread of the resulting output sample, which is measured, e.g., by quantile values, describes the ensemble's predictive uncertainty.

Such model ensembles can be constructed in different ways: a simple approach is to consider L models from the same model family (e.g., neural networks with the same architecture) that are optimized, independently of one another, on the entire training dataset. The resulting models (and thus their respective output values) differ from one another due to randomness in the optimization process (e.g., due to random model parameter initializations and the random composition and ordering of the training mini-batches), which entails convergence to qualitatively different minima. Among the more advanced ensembling approaches, two techniques are particularly important: *bootstrap aggregation* (bagging, see, e.g., Section 14.2 of Bishop (2006)), where models are trained on randomly drawn subsamples of the full training dataset, and *boosting* (ibid., Section 14.3), which relies on "ordered" model training where successive models focus on parts of the training data that were challenging for the previously trained models. This way, boosting combines models with (potentially) high biases such that the resulting ensemble exhibits low bias, whereas bagging yields a low-variance ensemble starting from models with high variances.

In deep learning, so-called deep ensembles (Lakshminarayanan et al., 2017) are widely used. For regression tasks, these ensembles typically build on "Gaussian" networks, i.e., networks whose outputs are interpreted as mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ of a Gaussian distribution. Assuming that all trained "Gaussian" networks contribute equally to the ensemble, the output distribution of the latter is given by the Gaussian mixture

$$P(\boldsymbol{y} \,|\, \boldsymbol{x}) = \frac{1}{L} \sum_{l=1}^{L} \mathcal{N}\Big[\boldsymbol{\mu}_l(\boldsymbol{x}), \boldsymbol{\Sigma}_l(\boldsymbol{x})\Big](\boldsymbol{y}) \ , \tag{3.4}$$

where $l = 1, \ldots, L$ indexes the ensemble components. To reduce the complexity coming along with such multi-modal Gaussian mixture distributions, they are often approximated

by simple unimodal Gaussian distributions with matching first and second moments. Deep ensembles constructed in this way were shown to yield accurate uncertainty estimates (see, e.g., Snoek et al. (2019); Gustafsson et al. (2020)), rendering them a state-of-the-art method. Fort et al. (2019) attribute these observations to an ensemble's ability to capture the multi-modality of loss landscapes, which allows for providing diverse (and often well-generalizing) sets of solutions. Several variants of deep ensembles were put forward, among others, resource-efficient approaches like batch-ensembles (Wen et al., 2020) and "masksembles", (Durasov et al., 2021) as well as techniques where component architectures are varied as part of the optimization procedure, for instance, hyperparameter ensembles (Wenzel et al., 2020) and NES ensembles (Zaidi et al., 2021).

Moreover, ensembling-based techniques can be combined with Bayesian approaches like Gaussian processes as is done by so-called Bayesian non-parametric ensembles (Liu et al., 2019). They facilitate the attribution of predictive uncertainty to aleatoric and (different kinds of) epistemic uncertainty. This characterization allows for a more adequate handling of challenging inputs and provides information on whether encountered uncertainty is reducible, e.g., by increasing the size of the training dataset.

Another type of frequentist approach are so-called jackknife-based methods that build on jackknife resampling (Quenouille, 1956; Tukey, 1958), which helps (similarly to bootstrapping) to determine variances and biases of statistical estimates. For a dataset of size N, N so-called leave-one-out estimates of a parameter are calculated, each of which based on a dataset of size $N-1$ obtained from the original dataset by systematically removing a single data point. Jackknife+ (Barber et al., 2021), for instance, uses jackknife resampling to construct model prediction intervals and to provide so-called coverage guarantees which ensure that test data labels lie (with high probability) within these intervals. The guarantees given by Jackknife+ hold for a broad range of data distributions.

A naive application of such jackknife methods for uncertainty estimation in neural networks would typically involve the training of N neural networks on respective leave-one-out training datasets. As such a procedure is prohibitively expensive in most cases, approximations of the leave-one-out predictions were proposed, for instance, by means of smaller ensembles (Kim et al., 2020) or based on techniques that build on the training loss incurred when optimizing a neural network on the full dataset (Alaa & Van Der Schaar, 2020). The output samples that these approximations yield for a given input are then used to quantify uncertainties by means of confidence intervals.

3.3. Evaluating uncertainty estimates

Evaluating uncertainty estimates is about comparing (often high-dimensional) probability distributions. Typically, it is assessed how well a predictive distribution, induced by uncertainty modeling techniques like the ones discussed above, matches a ground truth or reference distribution that reflects the given data-intrinsic uncertainty or the task performance of the model. Such quality assessments of predicted uncertainties are especially relevant in light of the (often strong) distributional and independence assumptions they

rely on, particularly, as the impact of these assumptions on the predicted uncertainties is difficult to quantify for complex model classes. While comparisons between ground truth and predicted distributions at the level of individual data points would be desirable, they are complicated by the (already mentioned) general unavailability of uncertainty ground truth information. Measures to evaluate the quality of uncertainty estimates vary, among others, in the way they circumvent this obstacle.

Negative log-likelihoods (NLLs), for instance, facilitate pointwise evaluations by providing information on how likely a ground truth value occurs given a model's output distribution (see, e.g., Blei & Jordan (2006); Walker et al. (2016); Gal & Ghahramani (2016a)). Motivated by the central limit theorem (CLT) and in accordance with modeling choices, the predicted output distributions are often assumed to be Gaussian for regression tasks.[1] As measures for likelihoods, NLL values are hybrid scores that entangle uncertainty quality with task performance and favor, given two models with comparable uncertainties, the better performing one. Moreover, they capture distributional properties of uncertainty estimates only in a limited fashion, as is discussed in Appendix B.4.2.

This is in contrast to calibration measures that put emphasis on distributional characteristics. They are set-based, i.e., statements w.r.t. calibration can, due to a lack of ground truth information, typically not be obtained for single data points but only for entire datasets or sufficiently large sub-sets of them. An important representative of this class is the *expected calibration error* (ECE, DeGroot & Fienberg (1983); Naeini et al. (2015)) that measures deviations between the distribution of predicted uncertainties and the distribution of the actually occurring model errors. While most contributions on ECEs focus on classification tasks as probabilistic classifiers naturally provide confidence information (see, e.g., DeGroot & Fienberg (1983); Guo et al. (2017)), ECE formulations for regression tasks also exist: following Kuleshov et al. (2018), an uncertainty estimator for a regression task is perfectly calibrated if, for each quantile value q, $q\%$ of all ground truth labels lie below the q-quantile values of the respectively predicted output distributions (that are, once again, typically assumed to be Gaussian). The regression ECE measures deviations from this ideal behavior. For reasons of technical feasibility, deviations are not determined for individual quantile values, but for quantile intervals that can be constructed in an overlapping or non-overlapping fashion. Aggregating the per-interval deviations then yields the regression ECE of a model. For further variants to calculate the ECE, see Roelofs et al. (2022) and references therein. While we focus on the ECE measure in this work (see Section 5.2 for technical details), alternative calibration scores exist. Gupta et al. (2021), for instance, propose a binning-free measure based on the Kolmogorov-Smirnov (KS) statistic. For detailed analyses and comparisons of different uncertainty measures (including a KS-based one), refer to Subsection 6.4.2 and Appendix B.4.1. To not only measure uncertainty calibration but to optimize for it as part of gradient-based model training, differentiable calibration measures (or approximations

[1] In some cases, related probability distributions are considered, deep evidential regression models, for instance, employ NLLs based on Student's t-distribution (see Section 3.2).

thereof) are required (compare Kumar et al. (2018), Krishnan & Tickoo (2020), and Karandikar et al. (2021)).

In particular for real-world applications, not only the calibration of uncertainty estimates is of interest but also their ability to solve downstream tasks, such as detecting out-of-distribution (OOD) inputs or adversarial attacks. Simple detection mechanisms can be obtained by thresholding uncertainty estimates and "raising a flag" when an uncertainty estimate surpasses a pre-defined threshold value. To evaluate the quality of such binary classifications independent of the particular choice of the threshold value, areas under the receiver operating characteristic and areas under the precision-recall curve are considered (see, e.g., Hendrycks & Gimpel (2017)).

Uncertainty estimators can moreover be assessed by investigating their qualitative behavior. Kendall & Gal (2017), for instance, analyze if the epistemic uncertainty decreases when increasing the size of the training dataset, and Wirges et al. (2019) study, for a 3D regression task, how uncertainty estimates depend on the distances of the detected objects. Visualizations[2], finally, can help to evaluate semantic high-level properties of uncertainty estimates that are difficult to quantify, as relevant semantic meta-information is generally not available.

[2]Think, for instance, of heatmaps that visualize pixel-level uncertainty estimates in the case of a semantic segmentation network.

4. Impact of Model Capacity on Uncertainty

Uncertainty modeling techniques seek, unlike their deterministic counterparts, not only to encode input-output dependencies but also to capture arising challenges, e.g., due to intrinsic data uncertainty or model class errors. Take, for instance, Fig. 3.3 (p. 30) as an illustrative example, where an under-complex probabilistic (neural) model is unable to closely approximate a high-frequency dataset. Instead, it roughly fits the data mean while predicting coarse fluctuation-averaged uncertainty intervals. These uncertainty estimates allow the model, figuratively speaking, to catch a glimpse of the more intricate actual input-output relations in the data. One may thus think of uncertainty estimation as an attempt to grasp the data properties, if not exactly, then at least in probability.

Starting from this observation, we study, in the present part of this work, how the quality and the properties of uncertainty estimates depend on model complexity. For low-dimensional toy datasets (see Fig. 3.3), inappropriate modeling choices are easily noticed, and more complex (e.g., deeper and wider neural) models may be chosen instead. The latter can express a broader range of input-output relations and are thus able to mimic, e.g., high-frequency data fluctuations. In this way, modeling residuals become small in magnitude and thus reduce the need to capture them using uncertainty estimates. However, for high-dimensional datasets (on which most real-world ML applications operate), such intuitive assessments of model complexity are not feasible, mainly because of their sparsity due to the "curse of dimensionality" (see, e.g., Section 1.4 of Bishop (2006)). Deterministic high-frequency relations (which could, in principle, be modeled) and data noise (that is irreducible) thus become virtually indistinguishable.

Technically, changes to the complexity of a model (with fixed input and output dimensions) can be made by manipulating its latent parameter space, for example, by varying its size (see Section 4.2) or by imposing correlations on it to effectively reduce model expressiveness (see Sections 4.1 and 4.2). Specifically, we study two types of models in this chapter: hidden Markov models (HMMs, see Section 4.1) and fully connected neural networks (FCNs, see Section 4.2). The former are simple models for uncertain dynamics of (in our case, discrete) sequential observations. These HMMs come along with an inherent notion of (inter-class) uncertainty. Thus, model quality and quality of uncertainty estimates are closely interlinked. The second model class of FCNs, in contrast, typically yields deterministic point estimates for regression setups and simple, often ill-calibrated confidence scores in the case of classification tasks. We extend FCNs to Bayesian networks by employing Monte Carlo (MC) dropout, a widely used regularization and uncertainty technique based on drawing random sub-networks during training and at inference (see Section 3.2 for details). For these MC dropout networks, we study how the distributional properties of their outputs depend on modeling choices, particularly when imposing

correlations on their hidden parameters. The output distributions carry information on both task performance and uncertainty quality: the mean values of these distributions are typically interpreted as the model's (mean) prediction (see Section 3.2), while uncertainty estimates are conveyed in their widths, which quantify how likely the mean value is actually realized. Dropout-enhanced neural models are, even beyond Section 4.2, a central object of study in this thesis, particularly in Chapter 5.

4.1. Capacity control of HMMs via representations

Hidden Markov models (Rabiner & Juang, 1986) have been a state-of-the-art approach for modeling sequential data for more than three decades (Hinton et al., 2012). Their success story is backed by a large number of applications ranging from natural language modeling (Chen & Goodman, 1999) and automatic speech recognition (Bahl et al., 1986) over information extraction (Scheffer et al., 2001) to robotics (Fu et al., 2016). While still being used frequently, many more recent approaches are based on neural networks instead, like feed-forward neural networks (Schmidhuber, 2015), recurrent neural networks (Hochreiter & Schmidhuber, 1997), or spiked neural networks (Tavanaei et al., 2019). However, the recent breakthroughs in the field of neural networks (Deng et al., 2013; LeCun et al., 2015; Schmidhuber, 2015; Paul et al., 2015; Goodfellow et al., 2016; Minar & Naher, 2018; Wu et al., 2020b) are accompanied by a substantial lack of their theoretical understanding. In contrast, HMMs come with a broad theoretical understanding, for instance, of the parameter estimation (Yang et al., 2017), convergence (Dempster et al., 1977; Wu, 1983), consistency (Leroux, 1992), and short-term prediction performance (Sharan et al., 2018), despite their non-convex optimization landscape.

Unlike HMMs, (neural) representation learning became more prominent only recently (Mikolov et al., 2013b; Pennington et al., 2014). From the first day following their release, approaches like word2vec or Glove (Mikolov et al., 2013a;b; Pennington et al., 2014; Le & Mikolov, 2014) that yield dense representations for state sequences, have significantly emphasized the value of pre-trained representations of discrete sequential data for downstream tasks (Kim, 2014; Wang et al., 2019). Since then, those found application not only in language modeling (Mikolov et al., 2013c; Zhang et al., 2016b; Li & Yang, 2018; Almeida & Xexéo, 2019) but also in biology (Asgari & Mofrad, 2015; Zou et al., 2019), graph analysis (Perozzi et al., 2014; Grover & Leskovec, 2016), and even banking (Baldassini & Serrano, 2018).

Ever since, the quality of representation models increased steadily, driven especially by the natural language community. Recently, so-called transformer networks (Vaswani et al., 2017) were put forward, complex deep architectures that leverage attention mechanisms (Bahdanau et al., 2015; Kim et al., 2017). Their complexity and tremendously large amounts of compute and training data led again to remarkable improvements on a multitude of natural language processing (NLP) tasks (Devlin et al., 2019).

These developments are particularly driven by the question on *how to optimally embed discrete sequences into a continuous space*. However, many existing approaches identify

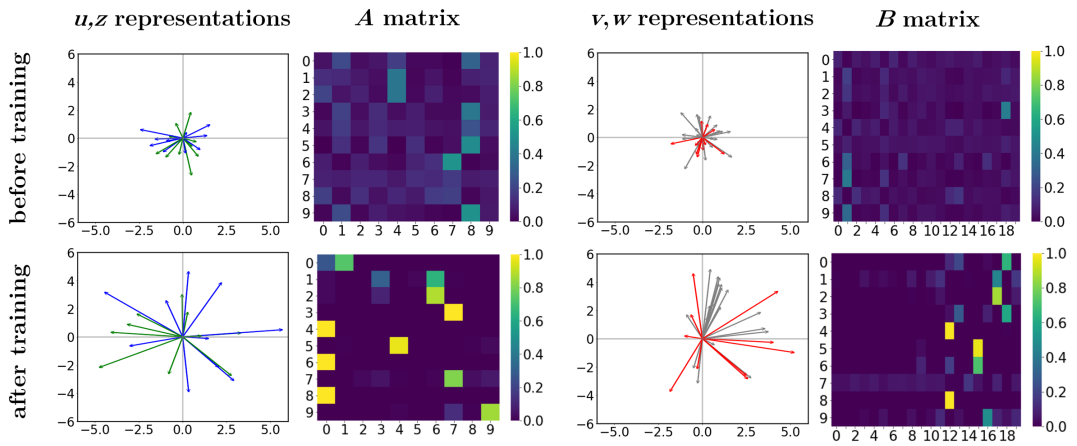


Fig. 4.1.: Exemplary DenseHMM to visualize the inner workings of our approach. All model components are shown before (top row) and after training (bottom row). The transition matrices $\boldsymbol{A}$ (second column) and $\boldsymbol{B}$ (fourth column) are learned by learning dense representations (first and third column). All representations are initialized by a standard Gaussian.

optimality solely with performance and put less emphasis on aspects like conceptual simplicity and theoretical soundness. Intensified discussions on well-understood and therefore trustworthy machine learning (Saltzer & Schroeder, 1975; Dwork et al., 2012; Amodei et al., 2016; Gu et al., 2017; Varshney, 2019; Toreini et al., 2020; Brundage et al., 2020) indicate, however, that these latter aspects become more and more crucial or even mandatory for real-world learning systems.

In light of this, we propose *DenseHMM* – a modification of hidden Markov models that allows us to learn dense representations of both the hidden states and the discrete observables (Fig. 4.1). Compared to the standard HMM, transition probabilities are not atomic but composed of these representations. Concretely, we contribute

- a parameter-efficient, nonlinear matrix factorization for HMMs,
- two competitive approaches to optimize the resulting DenseHMM
- and an empirical study of its performance and properties on diverse datasets.

The remainder of the section is organized as follows: first, we recapitulate established techniques for HMM optimization in Subsection 4.1.1 that the knowledgeable reader may skip. Our own contribution, DenseHMM and its optimization schemes are introduced in Subsection 4.1.2. We study the effect of its softmax nonlinearity and conduct empirical analyses and comparisons with standard HMMs in Subsections 4.1.3 and 4.1.4, respectively. A discussion in Subsection 4.1.5 concludes the section.

4.1.1. Recapitulation of HMM optimization

HMMs are generative models with Markov properties for sequences of either discrete or continuous observation symbols (Rabiner, 1989). They assume a number of non-observable (hidden) states that drive the dynamics of the generated sequences. If domain expertise allows for interpreting these drivers, HMMs can be fully understood. This distinguishes HMMs from sequence-modeling neural networks like long short-term memory networks (Hochreiter & Schmidhuber, 1997) and temporal convolutional networks (Bai et al., 2018). More recent latent variable models that keep the discrete structure of the latent space make use of, e.g., Indian buffet processes (Griffiths & Ghahramani, 2011). These enable a dynamic adaptation of the dimension of the latent space depending on data complexity and thus facilitate more flexible modeling. While we stay in the HMM model class, we argue that our approach allows for extending or reducing the latent space in a more principled way compared to standard HMMs.

Various approaches exist to learn the parameters of hidden Markov models: a classical one is the Baum-Welch algorithm (Rabiner, 1989) that handles the complexity of the joint likelihood of hidden states and observables by introducing an iterative two-step procedure that makes use of the forward-backward algorithm (Rabiner & Juang, 1986). Another algorithm for (local) likelihood maximization was proposed by Baldi & Chauvin (1994). Huang et al. (2018) study HMM learning on observation co-occurrences instead of observation sequences. Based on moments, i.e., co- and triple-occurrences, bounds on the empirical probabilities can be derived via spectral decomposition (Anandkumar et al., 2012). Approaches from Bayesian data analysis comprise Markov chain Monte Carlo (MCMC) and variational inference (VI). While MCMC can provide more stable and better results (Sipos, 2016), it traditionally suffers from poor scalability. A more scalable stochastic-gradient MCMC algorithm that tackles mini-batching of sequentially dependent data is due to Ma et al. (2017). The same authors propose a stochastic VI (SVI) algorithm (Foti et al., 2014) that shares some technical details with the work of Ma et al. (2017). SVI for hierarchical Dirichlet process (HDP) HMMs is considered by Zhang et al. (2016a). For our DenseHMM, we adapt two non-Bayesian procedures: the Baum-Welch algorithm (Rabiner, 1989) and direct co-occurrence optimization (Huang et al., 2018). The latter we optimize, solely for convenience, using a deep learning framework (see Appendix A.1.3 for details). Tran et al. (2016) carry this idea further, allowing different modifications to the original HMM context. Already earlier, combinations of HMMs and neural networks were employed, for instance for automatic speech recognition (Bourlard et al., 1992; Moon & Hwang, 1997; Trentin & Gori, 1999).

Non-negative matrix factorization (NMF, Lee & Seung (1999)) splits a matrix into a pair of low-rank matrices with solely positive components. NMF for HMM learning is used, e.g., by Lakshminarayanan & Raich (2010) and Cybenko & Crespi (2011). In contrast, we combine matrix factorization with a nonlinear kernel function to ensure non-negativity and normalization of the HMM transition matrices. Further foci of recent work on HMMs are identifiability (Huang et al., 2018), i.e., uniqueness guarantees for an obtained model, and optimized priors over transition distributions (Qiao et al., 2015).

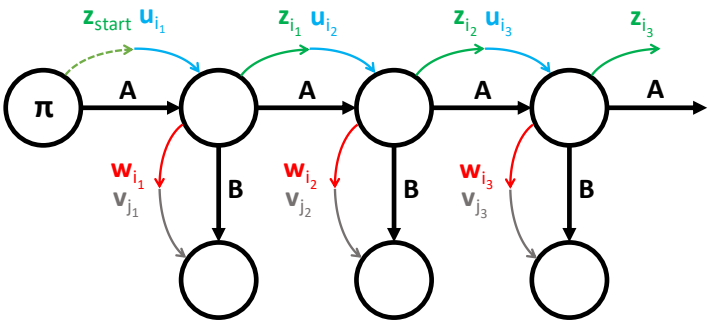


Fig. 4.2.: Structure of the DenseHMM. The HMM parameters $\mathbf{A}$, $\mathbf{B}$ and π are composed of vector representations such that $\mathbf{A} = \mathbf{A}(\mathbf{U}, \mathbf{Z})$, $\mathbf{B} = \mathbf{B}(\mathbf{V}, \mathbf{W})$ and $\pi = \pi(\mathbf{U}, \mathbf{z}_{\text{start}})$.

4.1.2. Structure and optimization of the DenseHMM

A HMM is defined by two time-discrete stochastic processes: $\{X_t\}_{t\in\mathbb{N}}$ is a Markov chain over hidden states $S = \{s_i\}_{i=1}^n$ and $\{Y_t\}_{t\in\mathbb{N}}$ is a process over observable states $O = \{o_i\}_{i=1}^m$ (see the parts of Fig. 4.2 that are displayed in black). The central assumption of HMMs is that the probability to observe $Y_t = y_t$ depends only on the current state of the hidden process and the probability to find $X_t = x_t$ only on the the previous state of the hidden process, $X_{t-1} = x_{t-1}$ for all $t \in \mathbb{N}$. We denote the state-transition matrix as $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ with $a_{ij} = P(X_t = s_j \,|\, X_{t-1} = s_i)$, the emission matrix as $\boldsymbol{B} \in \mathbb{R}^{n\times m}$ with $b_{ij} = P(Y_t = o_j \,|\, X_t = s_i)$ and the initial state distribution as $\boldsymbol{\pi} \in \mathbb{R}^n$ with $\pi_i = P(X_1 = s_i)$. A HMM is fully parameterized by $\boldsymbol{\lambda} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi})$.

HMMs can be seen as extensions of Markov chains that are in turn closely related to word2vec embeddings. Let us elaborate on this: a Markov chain has no hidden states and is defined by just one process $\{X_t\}_{t\in\mathbb{N}}$ over observables. The transition dynamics of the states of the Markov chain is described by a transition matrix $\boldsymbol{A}$ and an initial distribution $\boldsymbol{\pi}$. Being in a given state s_I, the Markov chain models conditional probabilities of the form $p(s_i \,|\, s_I)$. Markov chains are structurally similar to the approaches that learn word2vec representations, i.e., continuous bag of words and skip-gram (Mikolov et al., 2013b). Both models learn transitions between the words of a text corpus. Each word w_i of the vocabulary is represented by a learned dense vector $\boldsymbol{u}_i$. The transition probabilities between words are recovered from the scalar products of these vectors:

$$p(w_j \,|\, w_i) = \frac{\exp(\boldsymbol{u}_i \cdot \boldsymbol{v}_j)}{\sum_k \exp(\boldsymbol{u}_i \cdot \boldsymbol{v}_k)} \propto \exp(\boldsymbol{u}_i \cdot \boldsymbol{v}_j) \ . \tag{4.1}$$

The learned word2vec representations are low-dimensional and context-based, i.e., they contain semantic information. This is in contrast to the trivial and high-dimensional one-hot (or bag-of-word) encodings.

Here we transfer the nonlinear factorization approach of word2vec (Eq. 4.1) to HMMs. This is done by composing $\boldsymbol{A}, \boldsymbol{B}$ and $\boldsymbol{\pi}$ of dense vector representations such that

$$a_{ij} = a_{ij}(\mathbf{U}, \mathbf{Z}) = \frac{\exp(\boldsymbol{u}_j \cdot \boldsymbol{z}_i)}{\sum_{k\in[n]} \exp(\boldsymbol{u}_k \cdot \boldsymbol{z}_i)} \ , \tag{4.2a}$$

$$b_{ik} = b_{ik}(\mathbf{V}, \mathbf{W}) = \frac{\exp(\boldsymbol{v}_k \cdot \boldsymbol{w}_i)}{\sum_{k'\in[m]} \exp(\boldsymbol{v}_{k'} \cdot \boldsymbol{w}_i)} \ , \tag{4.2b}$$

$$\pi_i = \pi_i(\mathbf{U}, \mathbf{z}_{\text{start}}) = \frac{\exp(\boldsymbol{u}_i \cdot \boldsymbol{z}_{\text{start}})}{\sum_{k\in[n]} \exp(\boldsymbol{u}_k \cdot \boldsymbol{z}_{\text{start}})} \ , \tag{4.2c}$$

for $i, j \in [n]$ and $k \in [m]$. Let us motivate this transformation (Fig. 4.2) piece by piece: each representation vector corresponds to either a hidden state ($\boldsymbol{u}_i, \boldsymbol{w}_i, \boldsymbol{z}_i$) or an observation ($\boldsymbol{v}_i$). The vector $\boldsymbol{u}_i$ ($\boldsymbol{z}_i$) is the incoming (outgoing) representation of the hidden state i along the (hidden) Markov chain. The vector $\boldsymbol{w}_i$ is the outgoing representation of the hidden state i toward the observation symbols. These are described by the $\boldsymbol{v}_i$. All vectors are real-valued and of length l. $\boldsymbol{A}$ and $\boldsymbol{B}$ each depend on two kinds of representations instead of only one to enable non-symmetric transition matrices. Additionally, to choose $\boldsymbol{A}$ independent of $\boldsymbol{B}$, as is typical for HMMs, we need $\boldsymbol{w}_i$ as a third hidden representation. It is convenient to summarize all representation vectors of one kind in a matrix ($\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}, \boldsymbol{Z}$).

A softmax kernel maps the scalar products of the representations onto the HMM parameters $\boldsymbol{A}, \boldsymbol{B}$ and $\boldsymbol{\pi}$. Softmax maps to the simplex and thus ensures a_{ij}, b_{ij}, π_i to be in $[0, 1]$ as well as row-wise normalization of $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{\pi}$. While a different kernel function may be chosen, a strong nonlinearity, such as in the softmax kernel, is essential to obtain matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ with high ranks (compare experiments in Subsection 4.1.3).

This nonlinear kernelization enables constraint-free optimization, which is a central property of our approach. We use this fact in two different ways: first, we derive a modified expectation-maximization (EM) scheme (see next paragraph) and, second, study an alternative to EM optimization that is based on co-occurrences (see paragraph after next).

EM optimization with a gradient-based M-step We briefly recapitulate the EM-based Baum-Welch algorithm (see Subsection 13.2.2 of Bishop (2006)) and adapt it to learn the proposed representations as part of the M-step:

Given a sequence $\boldsymbol{o}$ of length $T \in \mathbb{N}$ over observations O, the Baum-Welch algorithm finds parameters λ that (locally) maximize the likelihood of the observations. A latent distribution Q over the hidden states S is introduced such that the log-likelihood of the sequence decomposes as follows:

$$\mathfrak{L}(Q, \boldsymbol{\lambda}) = \log P(\boldsymbol{o}, \boldsymbol{\lambda}) = \mathcal{L}(Q, \boldsymbol{\lambda}) + D_{\text{KL}}\Big(Q \,||\, P(\cdot \,|\, \boldsymbol{o}, \boldsymbol{\lambda})\Big) \ . \tag{4.3}$$

$D_{\mathrm{KL}}(P||Q)$ denotes the Kullback-Leibler divergence from Q to P with P and Q being probability distributions and

$$\mathcal{L}(Q, \boldsymbol{\lambda}) = \sum_{\boldsymbol{x} \in S^T} Q(\boldsymbol{x}) \log \frac{P(\boldsymbol{x}, \boldsymbol{o}; \boldsymbol{\lambda})}{Q(\boldsymbol{x})} \ . \tag{4.4}$$

Starting from an initial guess for $\boldsymbol{\lambda}$, the algorithm alternates between two sub-procedures, the E- and M-step: in the E-step, the forward-backward algorithm (see ibid.) is used to update $Q = P(\cdot \,|\, \boldsymbol{o}; \boldsymbol{\lambda})$, which maximizes $\mathfrak{L}(Q, \boldsymbol{\lambda})$ for fixed $\boldsymbol{\lambda}$. The efficient computation of the conditional probabilities $\gamma_t(s, s') := P(X_{t-1} = s, X_t = s' \,|\, \boldsymbol{o})$ and $\gamma_t(s) := P(X_t = s \,|\, \boldsymbol{o})$ for $s, s' \in S$ is crucial for the E-step. In the M-step, the latent distribution Q is fixed and $\mathfrak{L}(Q, \boldsymbol{\lambda})$ is maximized w.r.t. $\boldsymbol{\lambda}$ under normalization constraints. As the Kullback-Leibler divergence D_{KL} is set to zero in each E-step, the function to maximize in the M-step becomes

$$\mathcal{L}(Q, \boldsymbol{\lambda}) = \sum_{\boldsymbol{x} \in S^T} Q(\boldsymbol{x}) \log \frac{P(\boldsymbol{x}, \boldsymbol{o}; \boldsymbol{\lambda})}{Q(\boldsymbol{x})} = \sum_{\boldsymbol{x} \in S^T} P(\boldsymbol{x} \,|\, \boldsymbol{o}; \boldsymbol{\lambda}^{\mathrm{old}}) \log \frac{P(\boldsymbol{x}, \boldsymbol{o}; \boldsymbol{\lambda})}{P(\boldsymbol{x} \,|\, \boldsymbol{o}; \boldsymbol{\lambda}^{\mathrm{old}})} \tag{4.5}$$

with $\boldsymbol{\lambda}^{\mathrm{old}}$ being the parameter obtained in the previous M-step. Applying the logarithm to $P(\boldsymbol{x}, \boldsymbol{o}; \boldsymbol{\lambda})$, which has a product structure due to the Markov properties, splits the optimization objective into three summands. Each term depends on only one of the parameters $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi}$:

$$\boldsymbol{A}^*, \boldsymbol{B}^*, \boldsymbol{\pi}^* = \arg\max_{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi}} \mathcal{L}(Q, \boldsymbol{\lambda}) = \arg\max_{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi}} \mathcal{L}_1(Q, \boldsymbol{A}) + \mathcal{L}_2(Q, \boldsymbol{B}) + \mathcal{L}_3(Q, \boldsymbol{\pi}) \ . \tag{4.6}$$

Due to structural similarities between the three summands, we consider only $\mathcal{L}_1$ in the following. The treatment of $\mathcal{L}_2$ and $\mathcal{L}_3$ can be found in Appendix A.1.1. For $\mathcal{L}_1$, we have

$$\mathcal{L}_1(Q, \boldsymbol{A}) = \sum_{\boldsymbol{i} \in [n]^T} P(s_{\boldsymbol{i}} \,|\, \boldsymbol{o}; \boldsymbol{\lambda}^{\mathrm{old}}) \sum_{t=2}^{T} \log(a_{i_{t-1}, i_t}) \tag{4.7}$$

with multi-index $s_{\boldsymbol{i}} = s_{i_1}, ..., s_{i_T}$. The next step is to re-write $\mathcal{L}_1$ in terms of γ_t and to use Lagrange multipliers to ensure normalization. This gives the following part $\bar{\mathcal{L}}_1$ of the full Lagrangian $\bar{\mathcal{L}} = \bar{\mathcal{L}}_1 + \bar{\mathcal{L}}_2 + \bar{\mathcal{L}}_3$:

$$\bar{\mathcal{L}}_1 := \sum_{i,j \in [n]} \sum_{t=2}^{T} \gamma_t(s_i, s_j) \log a_{ij} + \sum_{i \in [n]} \varphi_i \Big(1 - \sum_{j \in [n]} a_{ij}\Big) \tag{4.8}$$

with Lagrange multipliers φ_i for each $i \in [n]$ to ensure that $\boldsymbol{A}$ is a proper transition matrix.

To optimize a DenseHMM, we leave the E-step unchanged and modify the M-step by applying the parameterization of $\boldsymbol{\lambda}$, i.e., Eq. 4.2a - 4.2c, to the Lagrangian $\bar{\mathcal{L}}$. Please note that we can drop all normalization constraints as they are explicitly enforced by the softmax function. This turns the original constrained optimization problem of the M-step into an

unconstrained one, leading to a Lagrangian of the form $\bar{\mathcal{L}}^{\text{dense}} = \bar{\mathcal{L}}_1^{\text{dense}} + \bar{\mathcal{L}}_2^{\text{dense}} + \bar{\mathcal{L}}_3^{\text{dense}}$ with

$$\bar{\mathcal{L}}_1^{\text{dense}} = \sum_{i,j\in[n]} \sum_{t=2}^{T} \gamma_t(s_i, s_j) \boldsymbol{u}_j \cdot \boldsymbol{z}_i - \sum_{i,j\in[n]} \sum_{t=2}^{T} \gamma_t(s_i, s_j) \log \sum_{k\in[n]} \exp(\boldsymbol{u}_k \cdot \boldsymbol{z}_i) \ . \tag{4.9}$$

We optimize $\bar{\mathcal{L}}^{\text{dense}}$ with gradient-descent procedures such as SGD (Bottou, 2010) and Adam (Kingma & Ba, 2015).

Direct optimization of observation co-occurrences Inspired by Huang et al. (2018), we investigate an alternative to the EM scheme: directly optimizing co-occurrence probabilities. The ground truth co-occurrences $\boldsymbol{\Omega}^{\text{gt}}$ are obtained from training data $\boldsymbol{o}$ by calculating the relative frequencies of subsequent pairs $(o_i(t), o_j(t+1)) \in O^2$. If we know that $\boldsymbol{o}$ is generated by a HMM with a stationary hidden process, we can easily compute $\boldsymbol{\Omega}^{\text{gt}}$ analytically as follows: we summarize all co-occurrence probabilities $\Omega_{ij} = P(Y_t = o_i, Y_{t+1} = o_j)$ for $i, j \in [m]$ in a co-occurrence matrix $\boldsymbol{\Omega} = \boldsymbol{B}^T \boldsymbol{\Theta} \boldsymbol{B}$ with $\Theta_{kl} = P(X_t = s_k, X_{t+1} = s_l)$ for $k, l \in [n]$. We can further write

$$\Theta_{kl} = P(X_{t+1} = s_l \,|\, X_t = s_k)\, P(X_t = s_k) = A_{kl}\, \pi_k \tag{4.10}$$

under the assumption that $\boldsymbol{\pi}$ is the stationary distribution of $\boldsymbol{A}$, i.e., $\pi_j = \sum_i A_{ij}\, \pi_i$ for all $i, j \in [n]$. Then, we obtain the co-occurrence probabilities

$$\Omega_{ij} = \sum_{k,l\in[n]} \pi_k\, b_{ki}\, a_{kl}\, b_{lj} \qquad \text{for } i, j \in [m] \ . \tag{4.11}$$

Parameterizing the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ according to Eq. 4.2a - 4.2b yields

$$\Omega_{ij}^{\text{dense}}(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}, \boldsymbol{Z}) = \sum_{k,l\in[n]} \pi_k\, b_{ki}(\mathbf{V}, \mathbf{W})\, a_{kl}(\mathbf{U}, \mathbf{Z})\, b_{lj}(\mathbf{V}, \mathbf{W}) \tag{4.12}$$

for $i, j \in [m]$. Please note that $\boldsymbol{\pi}$ is not parameterized here. Following our stationarity demand it is chosen as the eigenvector $\mathbf{v}_{\lambda=1}$ of $\boldsymbol{A}^T$. We minimize the squared distance between $\boldsymbol{\Omega}^{\text{dense}}$ and $\boldsymbol{\Omega}^{\text{gt}}$ w.r.t. the vector representations, i.e.,

$$\arg\min_{\boldsymbol{U},\boldsymbol{V},\boldsymbol{W},\boldsymbol{Z}} \ ||\boldsymbol{\Omega}^{\text{gt}} - \boldsymbol{\Omega}^{\text{dense}}(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{W}, \boldsymbol{Z})||_F^2 \ , \tag{4.13}$$

using gradient-descent procedures like SGD and Adam.

4.1.3. Impact of nonlinear kernelization

To further motivate our approach, please note that a standard HMM with n hidden states and m observation symbols has $n^2 + n\,(m-1) - 1$ degrees of freedom (DOFs), whereas a DenseHMM with representation length l has $l\,(3n + m + 1)$ DOFs. Therefore, a low-dimensional representation length l leads to DenseHMMs with fewer DOFs compared

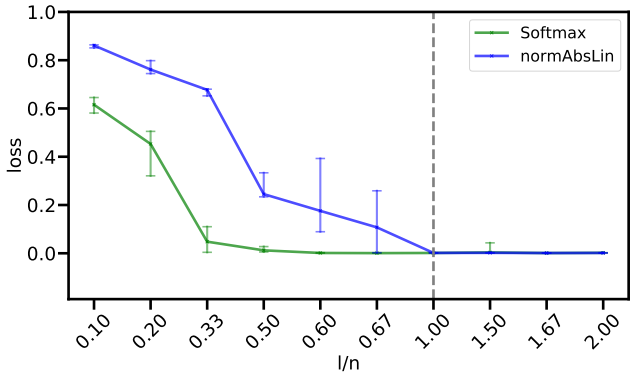


Fig. 4.3.: Approximation quality of nonlinear matrix factorizations. The optimization errors (median, 25/75 percentile) of softmax (green) and normAbsLin (blue) matrices are shown over the ratio of representation length l and matrix size n. The vertical line indicates $l = n$.

to a standard HMM for many values of n and m. A linear factorization with representation length $l < n$ leads to rank l for the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, whereas a nonlinear factorization can yield more expressive full rank matrices. This effect of nonlinearities may be best understood with a simple toy example: assume a 2×2 matrix with co-linear columns: $[[1, 2], [2, 4]]$. Applying a softmax column-wise leads to a matrix $\propto [[e, e^2], C[e^2, e^4]]$ with linearly independent columns (C denotes a constant). More general, the softmax rescales and rotates each column of $\boldsymbol{UZ}$ and $\boldsymbol{VW}$ differently and (except for special cases) thus increases the matrix rank to full rank. It is worth to mention that any other kernel $k : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R}^+$ could be used instead of the exponential function in the softmax to recover the HMM parameters from the representations, e.g., sigmoid, ReLU or radial basis functions (RBFs).

As nonlinear matrix factorization is a central building block of our approach, we compare the approximation quality of softmax with an appropriate linear factorization in the following setup: we generate a Dirichlet-distributed ground truth matrix $\boldsymbol{A}_{\text{gt}} \in \mathbb{R}^{n \times n}$ and approximate it (i) by $\tilde{\boldsymbol{A}} = \text{softmax}(\mathbf{UZ})$ defined by

$$(\tilde{\boldsymbol{A}})_{ij} = \text{softmax}(\mathbf{UZ})_{ij} = \frac{\exp\left((\mathbf{UZ})_{ij}\right)}{\sum_{k \in [n]} \exp\left((\mathbf{UZ})_{ik}\right)} \ , \tag{4.14}$$

and (ii) by a normalized absolute matrix product $\tilde{\boldsymbol{A}} = \text{normAbsLin}(\mathbf{UZ})$ defined by

$$(\tilde{\boldsymbol{A}})_{ij} = \text{normAbsLin}(\mathbf{UZ})_{ij} = \frac{|(\mathbf{UZ})_{ij}|}{\sum_{k \in [n]} |(\mathbf{UZ})_{ik}|} \ . \tag{4.15}$$

Note that we report the resulting error $||\tilde{\boldsymbol{A}} - \boldsymbol{A}_{\text{gt}}||_F$ divided by $||\boldsymbol{A}_{\text{gt}}||_F$ to get comparable losses independent of the size of $\boldsymbol{A}_{\text{gt}}$. These optimizations are performed for matrix sizes $n = 3, 5, 10$, several representation lengths l and 10 different $\boldsymbol{A}_{\text{gt}}$ for each (n, l) pair.

Tab. A.1 (p. 139) in Appendix A.1.3 provides all considered (n, l) pairs and detailed results. Fig. 4.3 shows that the softmax nonlinearity yields closer approximations of $\boldsymbol{A}_{\text{gt}}$ compared to normAbsLin. Moreover, we observe on a qualitative level significantly faster convergence for softmax as l increases. For softmax, vector lengths $l \approx n/3$ suffice to closely fit $\boldsymbol{A}_{\text{gt}}$ while the piecewise linear normAbsLin requires $l = n$. This result is in accordance with our remarks above.

4.1.4. Empirical evaluation

We investigate the outlined optimization schemes w.r.t. obtained model quality and behaviors. We compare the following types of models:

$\mathcal{H}_{\text{dense}}^{\text{EM}}$: a DenseHMM optimized with the EM optimization scheme (Subsection 4.1.2),

$\mathcal{H}_{\text{dense}}^{\text{direct}}$: a DenseHMM with directly optimized co-occurrences (Subsection 4.1.2),

$\mathcal{H}_{\text{stand}}$: a standard HMM optimized with the Baum-Welch algorithm (Rabiner, 1989).

These models all have the same number of hidden states n and observation symbols m. If a standard HMM and a DenseHMM use the same n and m, one of the models may have fewer DOFs than the other (see Subsection 4.1.3). Therefore, we also consider the model $\mathcal{H}_{\text{stand}}^{\text{fair}}$, which is a standard HMM with a number of DOFs that is comparable to a given $\mathcal{H}_{\text{dense}}$ model. We denote the number of hidden states in $\mathcal{H}_{\text{stand}}^{\text{fair}}$ as n_{fair}, which is the positive solution of

$$n_{\text{fair}}^2 + n_{\text{fair}}(m-1) - 1 = l\,(3n + m + 1) \tag{4.16}$$

rounded to the nearest integer. Note that n_{fair} can get significantly larger than n for $l > n$ and therefore $\mathcal{H}_{\text{stand}}^{\text{fair}}$ is expected to outperform the other models in these cases.

We use two standard measures to assess the model quality: the co-occurrence mean absolute deviation (MAD) and the normalized negative log-likelihood (NLL). The MAD between two co-occurrence matrices $\boldsymbol{\Omega}^{\text{gt}}$ and $\boldsymbol{\Omega}^{\text{model}}$ is defined as $1/m^2 \sum_{i,j\in[m]} |\Omega_{ij}^{\text{model}} - \Omega_{ij}^{\text{gt}}|$. We compute both $\boldsymbol{\Omega}^{\text{gt}}$ and $\boldsymbol{\Omega}^{\text{model}}$ based on sufficiently long sampled sequences (more details in Appendix A.1.3). In the case of synthetically generated ground truth sequences, we compute $\boldsymbol{\Omega}^{\text{gt}}$ analytically instead. In addition, we take a look at the negative log-likelihood of the ground truth test sequences $\{\boldsymbol{o}_i^{\text{test}}\}$ under the model, i.e., $\text{NLL} = -\sum_i \log P(\boldsymbol{o}_i^{\text{test}}; \boldsymbol{\lambda})$. We conduct experiments with $n \in \{3, 5, 10\}$ and different representation lengths l for each n. For each (n, l) combination, we run 10 experiments with different train-test splits. We evaluate the median and 25/75 percentiles of the co-occurrence MADs and the normalized NLLs for each of the four models (see Appendix A.1.3 for details).

In the following, we consider synthetically generated data as well as two real-world datasets: amino acid sequences from the RCSB PDB dataset (Berman et al., 2000) and part-of-speech tag sequences of biomedical text (Smith et al., 2004), referred to as the MedPost dataset.

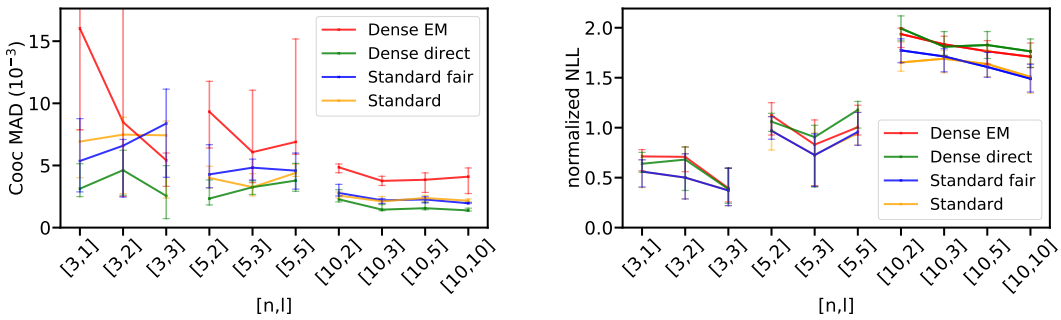


Fig. 4.4.: Co-occurrence mean absolute deviation (left) and normalized negative log-likelihood (right) of the models $\mathcal{H}^{\text{EM}}_{\text{dense}}$, $\mathcal{H}^{\text{direct}}_{\text{dense}}$, $\mathcal{H}^{\text{fair}}_{\text{stand}}$, $\mathcal{H}_{\text{stand}}$ on synthetically generated sequences evaluated for multiple combinations of n and l.

Synthetic sequences We sample training and test ground truth sequences from a standard HMM $\mathcal{H}_{\text{syn}}$ that is constructed as follows: each row of the transition matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ is drawn from a Dirichlet distribution $Dir(\boldsymbol{\alpha})$, where all entries in $\boldsymbol{\alpha}$ are set to a fixed value $\alpha = 0.1$. The initial state distribution $\boldsymbol{\pi}$ is set to the normalized eigenvector $\boldsymbol{v}_{\lambda=1}$ of $\boldsymbol{A}^T$. This renders $\mathcal{H}_{\text{syn}}$ stationary and allows a simple analytical calculation of $\boldsymbol{\Omega}^{\text{gt}}$ according to Eq. 4.11. For both training and testing, we sample 10 sequences, each of length 200 with $m = n$ emission symbols. Fig. 4.4 left shows our evaluation w.r.t. co-occurrence MADs. Note that the performance of $\mathcal{H}_{\text{stand}}$ changes slightly with l for fixed n as training is performed on different sequences for every (n, l) pair. This is because the sequences are re-drawn from $\mathcal{H}_{\text{syn}}$ for every experiment. The standard HMMs and $\mathcal{H}^{\text{direct}}_{\text{dense}}$ perform similarly, with $\mathcal{H}^{\text{direct}}_{\text{dense}}$ performing slightly better throughout the experiments and especially for $n = 3$. $\mathcal{H}^{\text{EM}}_{\text{dense}}$ shows a higher MAD than the other models. The good performance of $\mathcal{H}^{\text{direct}}_{\text{dense}}$ may be explained by the fact that it optimizes a function similar to co-occurrence MADs, whereas the other models aim to optimize negative log-likelihoods. The results in Fig. 4.4 right show that the DenseHMMs reach comparable NLLs, although the standard HMMs perform slightly better in this metric.

Proteins The RCSB PDB dataset (Berman et al., 2000) consists of 512,145 amino acid sequences from which we only take the first 1,024. After applying preprocessing (described in further detail in Appendix A.1.3), we randomly shuffle the sequences and split train and test data 50:50 for each experiment. Fig. 4.5 left shows the results of our evaluation w.r.t. co-occurrence MADs. $\mathcal{H}^{\text{EM}}_{\text{dense}}$ performs slightly worse than both $\mathcal{H}_{\text{stand}}$ and $\mathcal{H}^{\text{fair}}_{\text{stand}}$. We observe that $\mathcal{H}^{\text{direct}}_{\text{dense}}$ yields the best results. While the co-occurrence MADs of $\mathcal{H}^{\text{EM}}_{\text{dense}}$, $\mathcal{H}_{\text{stand}}$ and $\mathcal{H}^{\text{fair}}_{\text{stand}}$ stay roughly constant throughout different experiments, $\mathcal{H}^{\text{direct}}_{\text{dense}}$ can utilize larger n and l to further decrease co-occurrence MADs. As can be seen in Fig. 4.5 right, all models achieve almost identical normalized NLLs throughout the experiments. The results suggest that model size has only a minor impact on normalized NLL performance for the protein dataset.

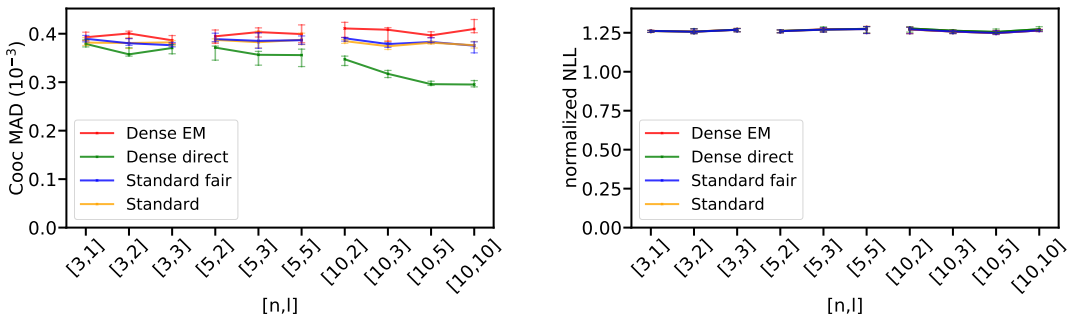


Fig. 4.5.: Co-occurrence mean absolute deviation (left) and normalized negative log-likelihood (right) of the models $\mathcal{H}_{\text{dense}}^{\text{EM}}$, $\mathcal{H}_{\text{dense}}^{\text{direct}}$, $\mathcal{H}_{\text{stand}}^{\text{fair}}$, $\mathcal{H}_{\text{stand}}$ on amino acid sequences evaluated for multiple combinations of n and l.

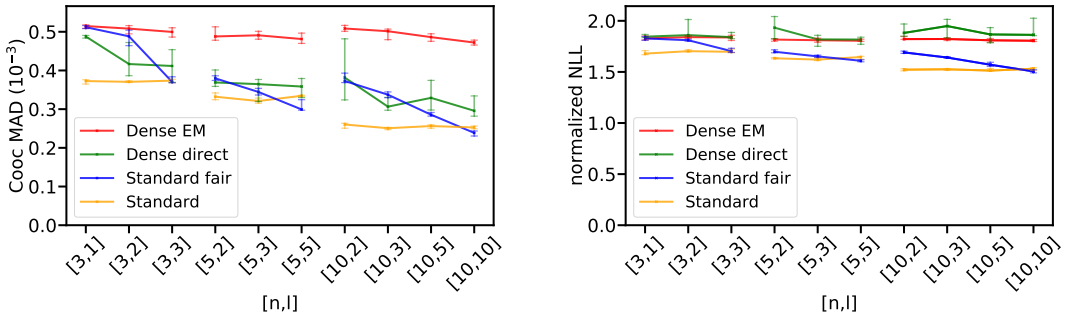


Fig. 4.6.: Co-occurrence mean absolute deviation (left) and normalized negative log-likelihood (right) of the models $\mathcal{H}_{\text{dense}}^{\text{EM}}$, $\mathcal{H}_{\text{dense}}^{\text{direct}}$, $\mathcal{H}_{\text{stand}}^{\text{fair}}$, $\mathcal{H}_{\text{stand}}$ on part-of-speech tag sequences (Medpost) evaluated for multiple combinations of n and l.

Part-of-speech sequences The MedPost dataset (Smith et al., 2004) consists of 5,700 sentences. Each sentence consists of words that were labeled with one of 60 part-of-speech tags. Sequences of part-of-speech tags are considered such that each sequence corresponds to one sentence. We apply preprocessing similar to the protein dataset (more details in Appendix A.1.3). The sequences are randomly shuffled and train and test data is split 50:50 for each experiment. Fig. 4.6 left shows performance of our models in terms of co-occurrence MADs. We see that the performance of the standard HMM models as well as $\mathcal{H}_{\text{dense}}^{\text{direct}}$ increases with increasing n. The number of hidden states seems to be a major driver of performance. Accordingly, $\mathcal{H}_{\text{stand}}^{\text{fair}}$ improves with growing $n_{\text{fair}}(l) \propto l$. Plus, we have $n_{\text{fair}} > n$ for $l \approx n$, which fully explains why $\mathcal{H}_{\text{stand}}^{\text{fair}}$ is the best performing model in these cases. Overall, $\mathcal{H}_{\text{dense}}^{\text{direct}}$ performs competitive to $\mathcal{H}_{\text{stand}}^{\text{fair}}$, especially for the practically more relevant cases with $l < n$. Similar to the other datasets, $\mathcal{H}_{\text{dense}}^{\text{EM}}$ performs worse than the other models and is barely affected by increasing n. Both DenseHMM models have increasing performance for increasing l. Normalized NLLs (Fig. 4.6 right) are best for

the standard HMM models. Both DenseHMM models achieve similar normalized NLLs, which are slightly worse than the ones achieved by the standard HMM models.

4.1.5. Discussion

We learn hidden Markov models by learning dense, real-valued vector representations for their n hidden and m observable states. The involved softmax nonlinearity enables to learn high-rank transition matrices and thus prevents that the matrix ranks are immediately determined by the chosen (in most cases) low-dimensional representation length l. In particular, we empirically find $l > n/3$ to be sufficient for closely approximating transition matrices of rank n (see Fig. 4.3). Fulfilling this condition allows for modeling a wide range of dynamics, whereas $l < n/3$ typically provides too coarse descriptions and thus indicates a low-quality modeling regime. Focusing on $l > n/3$, we successfully optimize DenseHMMs in two different ways and find direct co-occurrence optimization to yield competitive results compared to the standard HMM. This optimization technique requires only one gradient descent procedure and no iterative multi-step schemes. It is highly scalable with training data size and also with model size—as it is implemented in a modern deep learning framework. The optimization is stable and does neither require fine-tuning of learning rate nor of the representation initializations.[1]

We leave it to future work to adapt DenseHMMs to HMMs with continuous emissions and study variants of DenseHMM with fewer kinds of learnable representations. First experiments with DenseHMMs that learn only $\boldsymbol{Z}$ and $\boldsymbol{V}$ lead to almost comparable model quality. From a practitioner's viewpoint, it is worth investigating how DenseHMM and the learned representations perform on downstream tasks. Using the MedPost dataset, one could consider part-of-speech labeling of word sequences via the Viterbi algorithm after identifying the hidden states of the model with a pre-defined set of ground truth tags. For the protein dataset, a comparison with LSTM-based and BERT embeddings (Bepler & Berger, 2019; Min et al., 2019) could help to understand similarities and differences resulting from modern representation learning techniques. An analysis of geometrical properties of the learned representations seems promising for systems with $l \gg 1$ as $\exp(l)$ vectors can be almost-orthogonal in $\mathbb{R}^l$. The $\boldsymbol{V}$ representations are a natural choice for such a study as they directly correspond to observation items. The integration of Bayesian optimization techniques like MCMC and VI with DenseHMM is another research avenue.

4.2. Monte Carlo dropout in wide neural networks

To better understand the impact of model complexity on a model's ability to estimate uncertainty, we imposed correlations on the models' hidden state spaces in the previous section. Controlling their strengths by varying internal representation lengths, we identified regimes of differing model output quality after optimization. Here, we expand our analysis

[1] We release our tensorflow code to foster active use of DenseHMM in the community. It is available at `https://github.com/fraunhofer-iais/dense-hmm`.

of (dense) HMM models to the more complex class of fully connected networks (FCNs) with Monte Carlo dropout (see Section 3.2). Similar to the analyses before, we study the effect of correlations on the qualitative properties of the model output after, but in this case also before, optimization. For FCNs, weight correlations typically result from training and have an accumulating effect on inputs and features propagated through the network. To obtain a clearer picture of this effect, we, moreover, conduct investigations with simplified, manually set global parameter correlations. This accumulating effect is of interest as it can lead to deviation from otherwise Gaussian model output distributions which are often assumed to be the default case. This follows from connections between Gaussian processes (GPs) and Monte Carlo (MC) dropout networks in the limit of infinitely wide layers. We thus consider layer width as an additional control parameter in our experiments and investigate the prerequisites under which such limits of wide layers yield Gaussian and deviating non-Gaussian uncertainty distributions, respectively.

In Subsection 4.2.1, we take a closer look at requirements and boundary conditions established by previous works under which model output distributions are Gaussian. Next, we empirically study the pre-activations in randomly initialized and trained NNs with MC dropout in Subsection 4.2.2 and find that the former are (in accordance with theory) approximately normal distributed while the latter show a coexistence of Gaussian and non-Gaussian output distributions. Finally, we investigate how such deviations from Gaussian behavior accumulate from layer to layer using a toy model and NNs with strongly correlated weights (see Subsection 4.2.3).

4.2.1. Prerequisites for Gaussian pre-activation distributions

As a first step toward a characterization of NNs under MC dropout, we study a random feed-forward NN with $k+2$ layers. A *pre-activation* $f_i^{(\nu)}(\boldsymbol{x})$ in layer ν of this network is parameterized as follows:

$$f_i^{(\nu)}(\boldsymbol{x}) = \sum_{j=1}^{h_{\nu-1}(n)} \frac{\theta_{ij}^{(\nu)} g_j^{(\nu-1)}(\boldsymbol{x})}{\sqrt{h_{\nu-1}(n)}} + b_i^{(\nu)} \ , \tag{4.17}$$

where $g_i^{(\nu)}(\boldsymbol{x}) = z_i^{(\nu)} \phi(f_i^{(\nu)}(\boldsymbol{x}))$ for $\nu = 1, \ldots, k$, $g_i^{(0)}(\boldsymbol{x}) = x_i$ denote *activations* (after applying dropout) and $\boldsymbol{x} \in \mathbb{R}^d$ an input vector. Moreover, ϕ is the activation function, z_i are the dropout Bernoulli random variables with keep rate $q = 1 - p$, and $h_\nu(n)$ is the neuron count in layer ν. The learnable weights and biases of the network are termed $\theta_{ij}^{(\nu)}$ and $b_i^{(\nu)}$, respectively.

In previous work, see Matthews et al. (2018); Lee et al. (2018); Wu et al. (2019); Tsuchida et al. (2019), the authors considered NNs with independent prior distributions on the network parameters. The resulting summands in Eq. 4.17 are independent random variables such that the central limit theorem can readily be applied. In contrast, we examine a NN with a *fixed set* of parameters and dropout acting as an independent prior distribution on the activations. As a consequence, the resulting summands in Eq. 4.17 are

independent random variables only in the second layer ($\nu = 2$) and are dependent random variables for all successive ones. This follows from an inspection of the covariance of the pre-activations yielding

$$\begin{aligned} \mathrm{Cov}\left(f_i^{(\nu)}, f_j^{(\nu)}\right) = \frac{q}{h_{\nu-1}} \sum_{k,l=1}^{h_{\nu-1}(n)} \theta_{il}^{(\nu)} \theta_{jk}^{(\nu)} \\ \times \Big(\left(\delta_{kl} + q\ (1 - \delta_{kl})\right) \mathbb{E}_Z \left[\phi_k^{(\nu-1)} \phi_l^{(\nu-1)}\right] \\ - q\ \mathbb{E}_Z \left[\phi_k^{(\nu-1)}\right] \mathbb{E}_Z \left[\phi_l^{(\nu-1)}\right] \Big)\ , \end{aligned} \tag{4.18}$$

where we dropped the arguments of $f_i^{(\nu)}$ and $\phi_l^{(\nu-1)}$ for simplicity of notation. For finite $h_{\nu-1}(n)$ and $i \neq j$ the expression above is in general non-zero. Thus, we cannot apply the standard central limit theorem. However, the Gaussianity result of Matthews et al. (2018) can be extended to random networks with *fixed* independent weights by using self-averaging, see Section 2 and Appendix A of Sicking et al. (2020). Nevertheless, this theoretical insight cannot be easily transferred to trained NNs (of finite size) as the independence of weights can no longer be used. The empirical evaluations in the subsequent subsection indicate that non-Gaussianity in fact occurs, even for wide networks. For this reason, we focus our further investigation on the effects of strong correlations and their impact on tail behavior (see Subsection 4.2.3).

4.2.2. Ambiguous observations for empirical pre-activation distributions

To substantiate our discussions in Subsections 4.2.1 and 4.2.3 empirically, we study the pre-activations of a fully connected NN of the form given in Eq. 4.17. We train it for classification on FashionMNIST (Xiao et al., 2017) using a cross-entropy loss and hyperbolic tangent (tanh) nonlinearities. We employ a *narrow* network $\mathcal{H}_{\text{narrow}}$ with $k = 9$ hidden layers of width $h_\nu(n) = h = 100$ each as well as a *wide* network $\mathcal{H}_{\text{wide}}$ with $k = 7$ and $h = 1{,}000$. All network weights are initialized with i.i.d. random values. We train for 100 epochs using the Adam (Kingma & Ba, 2015) optimizer and an initial learning rate of $\mu = 0.001$ (see Appendix A.2.1 for more implementation details). Bernoulli dropout with $p = 0.2$ is applied to the activations of all hidden network layers.

First, we study the pre-activation distributions of the untrained $\mathcal{H}_{\text{narrow}}$ and $\mathcal{H}_{\text{wide}}$. These distributions are generated by running 30,000 forward passes with dropout for a fixed test image. We randomly pick one neuron from the last hidden layer of each network and visualize its pre-activation distribution in Fig. 4.7 (top row). To ease visual comparison between different neurons, we report normalized pre-activations. The resulting distributions are clearly Gaussian, see our discussion in Subsection 4.2.1. The weight correlations and pre-activation correlations of these untrained networks are studied in Appendix A.2.1.

Analyzing the trained networks $\mathcal{H}_{\text{narrow}}$ and $\mathcal{H}_{\text{wide}}$ yields more complex results. The pre-activation distributions for exemplarily selected hidden units from the last hidden

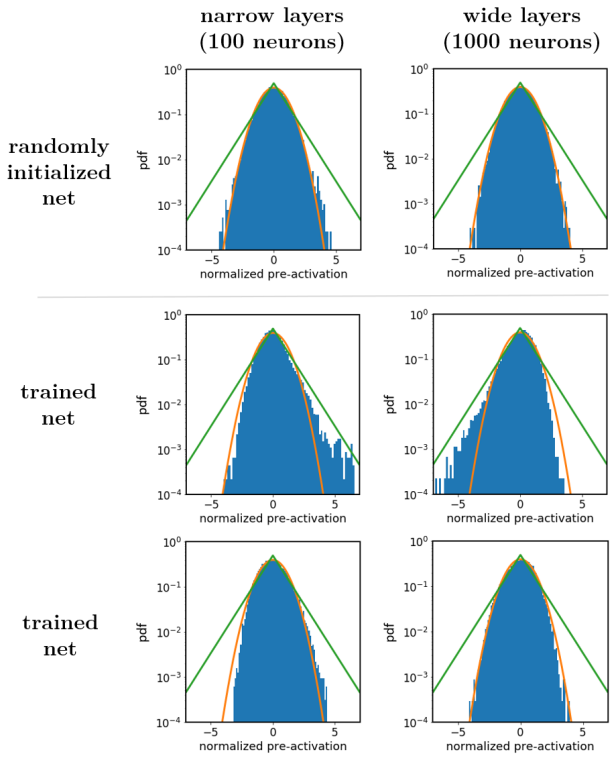


Fig. 4.7.: Normalized pre-activation distributions of selected neurons from different networks. We consider randomly initialized (top row) and trained (middle and bottom row) nets that are either narrow ($h = 100$, left column) or wide ($h = 1{,}000$, right column). We handpicked the neurons from the trained networks (see the text in Subsection 4.2.2). Standard normal (orange) and two-sided exponential (green) probability densities are given for comparison.

layer of the trained $\mathcal{H}_{\text{narrow}}$ and $\mathcal{H}_{\text{wide}}$ are shown in Fig. 4.7 (middle and bottom row). The neurons are handpicked to illustrate the variety of distributions we encounter in this layer. We observe Gaussian as well as non-Gaussian pre-activation distributions that range over skewed Gaussians to clearly non-Gaussian, exponential ones[2]. For $\mathcal{H}_{\text{wide}}$, approximately 40% of all neurons have Gaussian distributions, 40% skewed Gaussians and 20% exponential ones. Each histogram in Fig. 4.7 is based on 30,000 forward passes with dropout.

Moreover, we find these observations to depend on the input image: training inputs yield less tailed distributions than test inputs that in turn lead to less pronounced tails than artificial ones that are superpositions of two training images. However, there are large differences within those input categories, e.g., between test images. Shuffling the trained weight matrices yields results that are similar to the randomly initialized case. This indicates that weight and pre-activation dependencies are important while changes

[2]Technically also the tails of a Gaussian decay exponentially, however, with $\exp(-\xi^2)$. Throughout this text we refer to exponential tails only for the cases of $\exp(-|\xi|)$ or slower decay.

of the marginal distributions due to model training are not. While these results apply for the *last* hidden layer, we observe mostly Gaussians or slightly skewed Gaussians in earlier layers. In Subsection 4.2.3, we demonstrate on a toy model that non-Gaussian properties accumulate during the propagation through the network. Further empirical observations and an analysis of the weight correlations and pre-activation correlations of the trained networks can be found in Appendix A.2.1.

These complex dependence structures are the result of an interplay of weight distributions, input-immanent structures, and network nonlinearities that are orchestrated by network training. A theoretical foundation for the Gaussian results was laid out in the previous subsection. To address the deviations from Gaussian behavior, we take a closer look at correlations in the next subsection.

4.2.3. Modeling of strongly correlated systems

As we have seen in Eq. 4.18 the (pre-)activations from any layer ν are, in general, not independent from one another. We explore possible consequences of this observation in terms of consecutive toy experiments. For this, the product of *two* random variables, $Z = XY$, is central. On an abstract level, Y represents a random activation from a previous layer and X the product $w_{ij}z_j$. For later simplicity, we model both terms as Gaussian, $X, Y \sim \mathcal{N}(\mu, \sigma)$. In the case of vanishing mean, $\mu = 0$, one obtains the well known analytic form for the probability density function (PDF) of Z,

$$\mathrm{PDF}_Z(\xi) = \frac{1}{\pi} K_0(|\xi|) \qquad (\sigma = 1) \ , \tag{4.19}$$

with the modified Bessel function K_0. Derivations and further comments are relegated to Appendix A.2.2. In contrast to the normal distributions its asymptotic,

$$\mathrm{PDF}_Z(\xi) \sim \frac{1}{\sqrt{2\pi|\xi|}} \mathrm{e}^{-|\xi|} \qquad (\sigma = 1,\ |\xi| \gg 1) \ , \tag{4.20}$$

reveals exponentially decaying tails.

For any given neuron we encounter sums over h terms, where h denotes the layer width, instead of single products. If these summands X_iY_i were independent as argued for in Subsection 4.2.1, one would recover a normal distributed outcome for the neuron by the central limit theorem. But if they are correlated, the result may differ drastically. The outcome strongly depends on the respective covariance matrices of X_i, Y_i, for details see Appendix A.2.2. As a rule of thumb, larger correlations favor non-Gaussian results. This can be illustrated in terms of a toy extension choosing

$$Z = \sum_{i=1}^{h} X_i Y_i \,, \quad X_i = c\, x_0 + (1-c)\, x_i \tag{4.21}$$

and similarly for Y_i with Gaussian random variables x_γ, y_γ for $\gamma = 0, 1, \dots h$. The factor c controls a global correlation among the entries. For this case we find the decomposition

$$\sum_{i=1}^{h} X_i Y_i = (1-c)^2 \sum_{i=1}^{h} x_i\, y_i + c^2 h\, x_0\, y_0 + c\,(1-c) \left(x_0 \sum_{i=1}^{h} y_i + y_0 \sum_{i=1}^{h} x_i \right) . \tag{4.22}$$

Heuristically speaking, the first term is of Gaussian nature while the second (and successive ones) contribute exponential tails, see the limits for $c = 0$ or 1, respectively. An important observation is that both terms are on similar footing with respect to h, i.e., also in the large h limit the Gaussian term will not suppress the first one. This finding is due to the choice of a global correlation present in all h terms of x_i and y_i.

Assuming that the input Y has exponential tails, we can investigate how those propagate for the $Z = XY$ model (given $\mu_x = 0$). For this, we take

$$\mathrm{PDF}_Y(\xi) \propto \frac{1}{|\xi|^{(n-1)/n}} \exp\left(-\alpha |\xi|^{2/n}\right), \qquad n \in \mathbb{N}\,, \tag{4.23}$$

as an analytically tractable approximation capturing the tail behavior. For $n = 1$, i.e., Gaussian-like decay, we obtain a Bessel result for Z comparable to Eq. 4.19, in which the tail is modeled by $n = 2$, see Eq. 4.20. More generally, for small values of n we obtain the explicit asymptotics for Z as

$$\mathrm{PDF}_Z(\xi) \sim \frac{1}{|\xi|^{n/(n+1)}} \exp\left(-\kappa^{1/(n+1)} |\xi|^{2/(n+1)}\right) \tag{4.24}$$

with $|\xi| \gg 1$ and some positive constant $\kappa > 0$ depending on α^n/σ_X^2. As the decay slows down with each iteration, this effect might contribute to an "accumulation" of tails.

Inspired by the presented heuristics, we revisit the randomly initialized NN introduced in Subsection 4.2.2. However, we initialize the weight matrices in a *correlated* fashion similar to the toy model in Eq. 4.21, see Appendix A.2.2. The resulting distributions for the pre-activations given random input are shown in Fig. 4.8. With $c = 0.1$ the correlation is deliberately small and the dropout rate $p = 0.2$ set as before, but we investigate deeper networks with 50 layers. The left panel shows that for $h = 1{,}000$ only the initial layer follows a Gaussian behavior and all successive ones exhibit exponential tails of increasing strength. A closer look reveals that the decay, especially for later layers, is slightly weaker than exponential, which might be caused by the effects presented in Eq. 4.24. On the right hand side we compare the pre-activations of the 13th layer for different network widths, from shallow ($h = 100$) to wide ($h = 2{,}000$). Except for small fluctuations of the tails, which besides statistics are caused by choosing different (fixed) random inputs for each network, the result is stable with h, suggesting that also the infinite width limit will not lead to a Gaussian outcome.

Figures A.1 and A.2 (p. 140) in Appendix A.2.1 show the empirical correlations of the trained network from Subsection 4.2.2. While their structure is more involved, we find that the correlation of the pre-activations increases for deeper layers. More importantly,

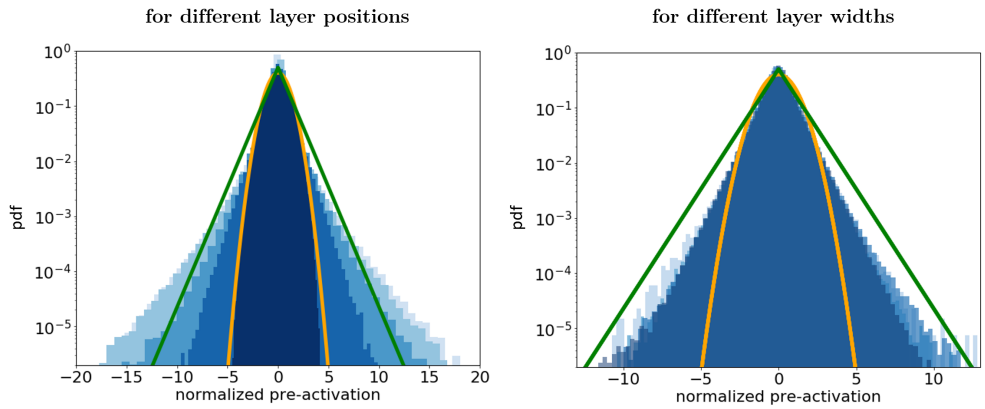


Fig. 4.8.: Normalized pre-activation distributions from networks with correlated random weights; for details, see the text in Subsection 4.2.3. On the l.h.s., we color-code hidden layer position, first (dark blue) to last (light blue). The r.h.s. shows different network widths (narrow (light blue) to wide (dark blue)) for layer 13. Gaussian (orange) and exponentially tailed distribution (green) are shown for comparison.

both correlations do not decrease to zero with increased layer width, which satisfies a central assumption of the toy model here. It might therefore be less surprising that the trained network can exhibit exponentially tailed pre-activations (see Fig. 4.7).

4.2.4. Discussion

First, we analyze neural networks with dropout and discuss independence assumptions as a prerequisite for their convergence to Gaussian processes in the limit of infinitely wide layers. Next, we empirically study wide trained networks. Unlike random networks, they exhibit a rich dependence structure and reveal a more complex picture: the coexistence of Gaussian and non-Gaussian exponential latent distributions. Finally, we shed light on this observation using a simple toy model and a network with correlated random initialization. These two systems indicate the existence of (at least) two regimes: one of *weakly* correlated pre-activations with Gaussian distributions and another one of *strongly* correlated pre-activations with exponentially tailed limiting distribution functions. Future work may investigate these two regimes in more detail to understand how to deliberately manipulate the properties of these limiting distributions under MC dropout. Searching for the borders of these regimes and further classes of limiting distributions is another research path. Our empirical observation of Gaussianity, which decreases from training data over test data to out-of-distribution data, gives rise to both theoretical and applied future work: first, to better understand the generalization properties of the *learned* Gaussian or non-Gaussian behavior and, second, to employ this knowledge to construct more robust uncertainty quantifications.

5. Modeling Uncertainty Estimates by Means of Wasserstein Dropout

In the previous chapter, we investigated the assumptions underlying MC dropout and constructed an example that violates the commonly assumed Gaussianity of output distributions. In this way, we demonstrated that the distributional properties of dropout-induced outputs can be influenced and steered. Moreover, we studied how model training impacts output and latent space distributions and found that 80% of the neurons in these networks still exhibit standard Gaussian behavior. These observations are relevant for two reasons: on the one hand, error sources are commonly modeled by Gaussians, which have the beneficial property that their quantile distributions are fully determined by (mean value μ and) standard deviation σ.[1] Having shown, on the other hand, that dropout-induced quantile distributions can be influenced, it seems feasible to also control their standard deviations in the Gaussian regime.

In this chapter, we propose a correspondingly built mechanism to adjust these local standard deviations, such that uncertainty estimates are better calibrated. Our mechanism is solely based on neuron dropout and therefore fully non-parametric[2], which is in contrast to widely used other techniques, compare Kendall & Gal (2017) and Lakshminarayanan et al. (2017), which combine their respective approaches with parametric predictions (see Section 3.2). Recalling the deviations from Gaussian behavior in the previous chapter, we stress that our approach to encode uncertainty is not limited to the Gaussian case but could, for instance, be adapted to the exponential classes considered before.

Concretely, we put forward *Wasserstein dropout (W-dropout)*, which is designed to capture heteroscedastic data noise via its sub-network distribution.[3] It builds on the idea of matching the network output distribution, resulting from randomly dropping neurons, to the (factual or implicit) data distribution by minimizing the Wasserstein distance. In detail, we contribute

- a novel and surprisingly simple Wasserstein-based learning objective for sub-networks that simultaneously optimizes task performance and uncertainty quality,

[1] The well-known 68-95-99.7 rule, for example, states the percentages of values covered when considering the output intervals $[\mu - n\,\sigma, \mu + n\,\sigma]$ of a Gaussian distribution for $n = 1, 2, 3$.

[2] While the term "non-parametric" can have multiple (related) meanings in different contexts of statistical modeling, in this thesis, we refer to quantities, in particular output distributions, that are not directly parameterized, e.g., by μ and σ for Gaussians, but obtained by sampling.

[3] The code base of Wasserstein dropout is publicly available at `https://github.com/fraunhofer-iais/wasserstein-dropout`.

- an extensive empirical evaluation where W-dropout outperforms state-of-the-art uncertainty techniques w.r.t. various benchmark metrics, not only in-data but also under data shifts,
- and two novel uncertainty measures: a non-saturating calibration score and a measure for distributional tails that allows us to analyze worst-case scenarios w.r.t. uncertainty quality.

The remainder of this chapter is organized as follows: first, Wasserstein dropout is introduced in Section 5.1. Its qualitative properties are analyzed and compared to MC dropout. Next, two novel uncertainty measures are proposed that address lacks of sensitivity of existing measures (Section 5.2). We then benchmark the uncertainties induced by Wasserstein dropout on various toy and standard datasets in Section 5.3, paying special attention to safety-relevant evaluation schemes and metrics. Finally, W-dropout is adapted to the complex task of object detection in Section 5.4, using (mostly) the compact SqueezeDet architecture (Wu et al., 2017). A summary of the obtained results and an outlook is provided in Section 5.5.

5.1. Motivation and derivation

Before we lay out our dropout-based approach to modeling *aleatoric* uncertainty, we analyze some central properties of Monte Carlo dropout. The latter also employs sub-networks, however, for the purpose of modeling *epistemic* uncertainty (Gal & Ghahramani, 2016a): Given a neural network $f_{\boldsymbol{\theta}} : \mathbb{R}^{n_{\text{in}}} \to \mathbb{R}^{n_{\text{out}}}$ with parameters $\boldsymbol{\theta}$, MC dropout samples sub-networks $f_{\tilde{\boldsymbol{\theta}}}$ by randomly dropping nodes from the main model $f_{\boldsymbol{\theta}}$ yielding for each input x_i a distribution $\mathcal{D}_{\tilde{\boldsymbol{\theta}}}(x_i)$ over network predictions. During MC dropout inference the final prediction is given by the mean of a sample from $\mathcal{D}_{\tilde{\boldsymbol{\theta}}}(x_i)$, while the uncertainty associated with this prediction can be estimated as a sum of its variance and a constant uncertainty offset. The value of the latter term requires dataset-specific optimization. During MC dropout training, minimizing the objective function, e.g., the mean squared error (MSE), shifts all sub-network predictions toward the same training targets. For a more formal explanation of this behavior, and without loss of generality, let $f_{\boldsymbol{\theta}}$ be a NN with one-dimensional output. The expected MSE for a training sample (x_i, y_i) under the model's output distribution $\mathcal{D}_{\tilde{\boldsymbol{\theta}}}(x_i)$ is given by

$$\mathbb{E}_{\tilde{\boldsymbol{\theta}}}\left[\left(f_{\tilde{\boldsymbol{\theta}}}(x_i) - y_i\right)^2\right] = \left(\mu_{\tilde{\boldsymbol{\theta}}}(x_i) - y_i\right)^2 + \sigma^2_{\tilde{\boldsymbol{\theta}}}(x_i) \ , \tag{5.1}$$

with sub-network mean $\mu_{\tilde{\boldsymbol{\theta}}}(x_i) = \mathbb{E}_{\tilde{\boldsymbol{\theta}}}[f_{\tilde{\boldsymbol{\theta}}}(x_i)]$ and variance $\sigma^2_{\tilde{\boldsymbol{\theta}}}(x_i) = \mathbb{E}_{\tilde{\boldsymbol{\theta}}}[f^2_{\tilde{\boldsymbol{\theta}}}(x_i)] - \mathbb{E}_{\tilde{\boldsymbol{\theta}}}[f_{\tilde{\boldsymbol{\theta}}}(x_i)]^2$. Therefore, training simultaneously minimizes the squared error between sub-network mean $\mu_{\tilde{\boldsymbol{\theta}}}(x_i)$ and target y_i as well as the variance $\sigma^2_{\tilde{\boldsymbol{\theta}}}(x_i)$.

As we, in contrast, seek to employ sub-networks to model *aleatoric* uncertainty, minimizing the variance over the sub-networks is not desirable for our purpose. Instead, we aim at explicitly fitting the sub-network variance $\sigma^2_{\tilde{\boldsymbol{\theta}}}(x_i)$ to the input-dependent, i.e., heteroscedastic, data variance. That is to say, we not only match the mean values as in

Eq. 5.1 but seek to match the *entire* data distribution $\mathcal{D}_y(x_i)$ by means of the model's output distribution $\mathcal{D}_{\tilde{\boldsymbol{\theta}}}(x_i)$. This output distribution is induced by applying Bernoulli dropout to all activations of the network. The matchings are technically realized by minimizing a distance measure between the two distributions $\mathcal{D}_{\tilde{\boldsymbol{\theta}}}(x_i)$ and $\mathcal{D}_y(x_i)$. While, in principle, various distances could be used, we, however, require two properties: i) the distance needs to be non-saturating, i.e., it needs to grow monotonously and unboundedly with the actual mismatch between the distributions. This is needed (or desirable) as for safety reasons, we want to penalize strong mismatches. Additionally, ii) we require the distance to have a simple, closed form. This is needed for subsequent, bootstrap-inspired approximations (see below). The (squared) 2-Wasserstein distance (Villani, 2008) fulfills both of these properties[4] and is therefore employed in the following. Assuming that both distributions $\mathcal{D}_{\tilde{\boldsymbol{\theta}}}(x_i)$ and $\mathcal{D}_y(x_i)$ are Gaussian[5] then yields a compact analytical expression

$$\begin{aligned} \mathrm{WS}_2^2(x_i) &= \mathrm{WS}_2^2\left[\mathcal{D}_{\tilde{\boldsymbol{\theta}}}(x_i), \mathcal{D}_y(x_i)\right] \\ &= \mathrm{WS}_2^2\left[\mathcal{N}(\mu_{\tilde{\boldsymbol{\theta}}}(x_i), \sigma_{\tilde{\boldsymbol{\theta}}}(x_i)), \mathcal{N}(\mu_y(x_i), \sigma_y(x_i))\right] \\ &= \left(\mu_{\tilde{\boldsymbol{\theta}}}(x_i) - \mu_y(x_i)\right)^2 + \left(\sigma_{\tilde{\boldsymbol{\theta}}}(x_i) - \sigma_y(x_i)\right)^2 , \end{aligned} \tag{5.2}$$

with $\mu_{\tilde{\boldsymbol{\theta}}}(x_i) = \mathbb{E}_{\tilde{\boldsymbol{\theta}}}[f_{\tilde{\boldsymbol{\theta}}}(x_i)]$ and $\sigma^2_{\tilde{\boldsymbol{\theta}}}(x_i) = \mathbb{E}_{\tilde{\boldsymbol{\theta}}}[(f_{\tilde{\boldsymbol{\theta}}}(x_i) - \mathbb{E}_{\tilde{\boldsymbol{\theta}}}[f_{\tilde{\boldsymbol{\theta}}}(x_i)])^2]$, and μ_y, σ_y defined analogously w.r.t. the data distribution.

In practice, however, Eq. 5.2 cannot be readily used as the distribution of y given x_i is typically not accessible. Instead, for a given, fixed value of x_i from the training set only a single value of y_i is known. Therefore, we take y_i as a (rough) one-sample approximation of the mean $\mu_y(x_i)$ resulting in $\mu_y(x_i) \approx y_i$ and $\sigma_y^2(x_i) \approx \mathbb{E}_y[(y - y_i)^2]$. However, $\sigma_y^2(x_i)$ cannot be inferred from a single sample. Inspired by parametric bootstrapping (Dekking et al., 2005; Hastie et al., 2009), we therefore approximate the empirical data variance (for a given mean value y_i and input x_i) with samples from our model, i.e., we approximate $\mathbb{E}_y[(y - y_i)^2]$ by

$$\mathbb{E}_{\tilde{\boldsymbol{\theta}}}[(f_{\tilde{\boldsymbol{\theta}}}(x_i) - y_i)^2] = (\mu_{\tilde{\boldsymbol{\theta}}}(x_i) - y_i)^2 + \sigma^2_{\tilde{\boldsymbol{\theta}}}(x_i) \ . \tag{5.3}$$

Inserting our approximations $\mu_y(x_i) \approx y_i$ and $\sigma_y(x_i) \approx (\mu_{\tilde{\boldsymbol{\theta}}}(x_i) - y_i)^2 + \sigma^2_{\tilde{\boldsymbol{\theta}}}(x_i)$ into Eq. 5.2 yields the *Wasserstein dropout loss* (W-dropout) for a data point (x_i, y_i) from the training distribution,

$$\mathrm{WS}_2^2(x_i) \approx (\mu_{\tilde{\boldsymbol{\theta}}}(x_i) - y_i)^2 + \left[\sqrt{\sigma^2_{\tilde{\boldsymbol{\theta}}}(x_i)} - \sqrt{(\mu_{\tilde{\boldsymbol{\theta}}}(x_i) - y_i)^2 + \sigma^2_{\tilde{\boldsymbol{\theta}}}(x_i)}\right]^2 \ . \tag{5.4}$$

[4] This is in contrast to other widely used metrics. The Kolmogorov-Smirnov (KS) statistic, for example, is saturating and therefore violates the first requirement whereas the KL divergence possesses a more involved structure that violates the second requirement.

[5] An assumption shared by, e.g., the NLL optimization or the ECE. While different distributions, for example exponentially decaying or mixtures, could be used in principle, we restrict the scope here to this standard Gaussian case.

Considering a mini-batch of size M instead of a single data point, the optimization objective is given by the arithmetic mean of the corresponding Eq. 5.4 terms. In practice, $\mu_{\tilde{\boldsymbol{\theta}}}(x_i)$ and $\sigma^2_{\tilde{\boldsymbol{\theta}}}(x_i)$ are approximated by empirical estimators using a sample size L, i.e.,

$$\mu_{\tilde{\boldsymbol{\theta}}}(x_i) \approx \frac{1}{L}\sum_{l=1}^{L} f_{\tilde{\theta}_l}(x_i) \ , \tag{5.5}$$

$$\sigma^2_{\tilde{\boldsymbol{\theta}}}(x_i) \approx \frac{1}{L}\sum_{l=1}^{L} f^2_{\tilde{\theta}_l}(x_i) - \left(\frac{1}{L}\sum_{l=1}^{L} f_{\tilde{\theta}_l}(x_i)\right)^2 \ . \tag{5.6}$$

In contrast to MC dropout we require thereby L stochastic forward passes per data point during training (instead of one), while at inference the procedures are exactly the same.

Besides the regression tasks considered here, our approach could be useful for other objectives that use or benefit from an underlying distribution, e.g., Dirichlet distributions to quantify uncertainty in classification, as discussed in the conclusion of this chapter.

An alternative formulation of Wasserstein dropout grounded in error learning While the approximations of Eq. 5.2 presented above are well-motivated, plausible alternative approaches exist. In the following, we sketch a different approximation of Eq. 5.2 that makes use of conceptual overlaps between uncertainty modeling and error learning.

To minimize the Wasserstein distance between the model output distribution and data distribution (that are both assumed to be Gaussian), the model's mean prediction $\mu_{\tilde{\boldsymbol{\theta}}}(x_i)$ and standard deviation $\sigma_{\tilde{\boldsymbol{\theta}}}(x_i)$ need to be optimized concurrently. As both model outputs, $\mu_{\tilde{\boldsymbol{\theta}}}(x_i)$ and $\sigma_{\tilde{\boldsymbol{\theta}}}(x_i)$, are parameterized by a shared set of global parameters (e.g., of a neural network), trade-off relations may occur between them. For Gaussian parameters, these trade-off relations can be quantified by means of the Gaussian negative log-likelihood,

$$\mathrm{NLL}_{\mathrm{normal}} \propto \log \sigma_{\tilde{\boldsymbol{\theta}}}(x_i) + \frac{(\mu_{\tilde{\boldsymbol{\theta}}}(x_i) - y_i)^2}{2\,\sigma^2_{\tilde{\boldsymbol{\theta}}}(x_i)} \ . \tag{5.7}$$

Assuming, for instance, a discrete label y_i (i.e., a label distribution $p(y \,|\, x_i) = \delta(y_i)$) and a model error $|\mu_{\tilde{\boldsymbol{\theta}}}(x_i) - y_i| > 0$, the NLL is optimized for a standard deviation that equals this error, $\sigma_{\tilde{\boldsymbol{\theta}}}(x_i) = |\mu_{\tilde{\boldsymbol{\theta}}}(x_i) - y_i|$. Thinking instead of y_i as a realization of a continuous label distribution $p(y \,|\, x)$, the (error) term $|\mu_{\tilde{\boldsymbol{\theta}}}(x_i) - y_i|$ can alternatively be seen as a rough proxy for (the scale of) the intrinsic uncertainty in the data distribution when assuming that the model's mean prediction $\mu_{\tilde{\boldsymbol{\theta}}}(x_i)$ approximates the mean of the data distribution. It therefore seems plausible to consider $\mu_y(x_i) \approx y_i$ and $\sigma_y(x_i) \approx |\mu_{\tilde{\boldsymbol{\theta}}}(x_i) - y_i|$. Plugging these approximations into Eq. 5.2 yields for a data point (x_i, y_i) from the training distribution

$$\mathrm{WS}^2_{\mathrm{GL}}(x_i) \approx (\mu_{\tilde{\boldsymbol{\theta}}}(x_i) - y_i)^2 + \left(\sigma_{\tilde{\boldsymbol{\theta}}}(x_i) - |\mu_{\tilde{\boldsymbol{\theta}}}(x_i) - y_i|\right)^2 . \tag{5.8}$$

We refer to this objective as the *Gaussian-likelihood variant of the Wasserstein dropout loss* (GL-W-dropout). As before, all model outputs $\mu_{\tilde{\boldsymbol{\theta}}}(x_i)$ and $\sigma_{\tilde{\boldsymbol{\theta}}}(x_i)$ are approximated

by sampling. The mini-batch loss is again obtained by averaging over the respective per-data-point losses. As stated earlier, this approach to encode the deviation between $|\mu_{\tilde{\boldsymbol{\theta}}}(x_i) - y_i|$ and $\sigma_{\tilde{\boldsymbol{\theta}}}(x_i)$ is closely linked to the topic of error learning. Objective functions similar to Eq. 5.8 exist in the literature, see, e.g., Eq. 4 in Feng et al. (2019).

The Gaussian likelihood variant of W-dropout (Eq. 5.8) requires, like standard W-dropout (Eq. 5.4), L stochastic forward passes per input in each training step. While training compute is (for many applications) less critical compared to compute at inference, reductions of the former one are still desirable to render the Wasserstein dropout technique affordable for large industry-scale networks. As the encoding of uncertainty information into the spread of the sub-network output distribution requires the concurrent optimization of several sub-networks, we seek to encode uncertainty information in a computationally cheaper way. Instead of generating different outputs by imposing different dropout masks on the network, one may, alternatively, generate different outputs in a *binary* way by switching the stochasticity in the network "off" and "on", i.e., by considering the full deterministic network $f_{\boldsymbol{\theta}}(x_i)$ and *one* randomly drawn sub-network $f_{\tilde{\boldsymbol{\theta}}}(x_i)$ for each input data point. The deterministic network $f_{\boldsymbol{\theta}}(x_i)$ is used to fit the data mean whereas the sub-network $f_{\tilde{\boldsymbol{\theta}}}(x_i)$ is explicitly encouraged *not* to do so. Instead, sub-networks are employed to model aleatoric uncertainty and prediction residuals if the prediction of the full network $f_{\boldsymbol{\theta}}(x_i)$ is incorrect. This is done by encoding the error of the network, $|f_{\boldsymbol{\theta}}(x_i) - y_i|$, into the distance $|f_{\tilde{\boldsymbol{\theta}}}(x_i) - f_{\boldsymbol{\theta}}(x_i)|$ between deterministic network $f_{\boldsymbol{\theta}}(x_i)$ and stochastic sub-network $f_{\tilde{\boldsymbol{\theta}}}(x_i)$. Thus, we deliberately assign different "tasks" to the full network $f_{\boldsymbol{\theta}}(x_i)$, on the one hand, and its sub-networks $f_{\tilde{\boldsymbol{\theta}}}(x_i)$, on the other hand. Adapting Eq. 5.8 accordingly, i.e., replacing $\mu_{\tilde{\boldsymbol{\theta}}}(x_i)$ by $f_{\boldsymbol{\theta}}(x_i)$ and $\sigma_{\tilde{\boldsymbol{\theta}}}(x_i)$ by $|f_{\tilde{\boldsymbol{\theta}}}(x_i) - f_{\boldsymbol{\theta}}(x_i)|$ yields[6]

$$\mathrm{WS}^2_{\text{GL-OS}}(x_i) \approx (f_{\boldsymbol{\theta}}(x_i) - y_i)^2 + \left(|f_{\tilde{\boldsymbol{\theta}}}(x_i) - f_{\boldsymbol{\theta}}(x_i)| - |f_{\boldsymbol{\theta}}(x_i) - y_i|\right)^2 \ , \tag{5.9}$$

which is termed the Gaussian-likelihood *one-sample* (GL-OS) variant of the Wasserstein dropout objective. Its first term is the MSE of the full network $f_{\boldsymbol{\theta}}$. Its second term seeks to optimize[7] the sub-networks $f_{\tilde{\boldsymbol{\theta}}}$. It aims at finding sub-networks such that the distance $|f_{\tilde{\boldsymbol{\theta}}}(x_i) - f_{\boldsymbol{\theta}}(x_i)|$ matches the prediction residual $|f_{\boldsymbol{\theta}}(x_i) - y_i|$, which also serves as a proxy for the aleatoric uncertainty. As our choice of the second loss term removes all directional information of the residual, possible (optimal) solutions for the $f_{\tilde{\boldsymbol{\theta}}}(x_i)$ are not uniquely determined.[8] The standard deviations $\sigma_{\text{total}}(x_i)$ of the predictions of the sub-networks w.r.t. the prediction of the mean network have two components: the spread $\sigma_{\tilde{\boldsymbol{\theta}}}(x_i)$ of the sub-networks and an offset $\left|f_{\boldsymbol{\theta}}(x_i) - \mu_{\tilde{\boldsymbol{\theta}}}(x_i)\right|$ between the full network $f_{\boldsymbol{\theta}}(x_i)$ and the sub-network mean $\mu_{\tilde{\boldsymbol{\theta}}}(x_i)$ that our loss might cause. Concretely,

$$\sigma_{\text{total}}(x_i) = \sigma_{\tilde{\boldsymbol{\theta}}}(x_i) + \left|f_{\boldsymbol{\theta}}(x_i) - \mu_{\tilde{\boldsymbol{\theta}}}(x_i)\right| \ . \tag{5.10}$$

[6] Please note that calculating $|f_{\boldsymbol{\theta}}(x_i) - f_{\tilde{\boldsymbol{\theta}}}(x_i)|$ requires only one stochastic and one deterministic forward pass, whereas multiple stochastic forward passes are needed to determine $\sigma_{\tilde{\boldsymbol{\theta}}}(x_i)$.

[7] To avoid unintended optimization of the full network $f_{\boldsymbol{\theta}}$ in direction of $f_{\tilde{\boldsymbol{\theta}}}$, we only back-propagate through $f_{\tilde{\boldsymbol{\theta}}}$ in the second loss term.

[8] For an analytical study of the loss landscape induced by Eq. 5.9, see Appendix B.1.1.

While $|f_{\boldsymbol{\theta}}(x_i) - \mu_{\tilde{\boldsymbol{\theta}}}(x_i)|$ is reminiscent of residual matching, $\sigma_{\tilde{\boldsymbol{\theta}}}(x_i)$ is more closely related to modeling uncertainties. We show in Appendix B.1.2 that $\sigma_{\tilde{\boldsymbol{\theta}}}(x_i)$ accounts on average for more than 80% of $\sigma_{\text{total}}(x_i)$ in our experiments. Please recall that the calculation of $\sigma_{\text{total}}(x_i)$ for an input data point x_i requires one deterministic and multiple stochastic forward passes.[9] Further note that the one-sample approximation described above can not only be applied to the Gaussian-likelihood variant of Wasserstein dropout loss (Eq. 5.8) but also to the standard Wasserstein dropout loss (Eq. 5.4). Preliminary experiments, however, show that such a "one-sample" Wasserstein dropout loss (OS-W-dropout) leads to overestimated uncertainties as implicit regularization of $\sigma_{\tilde{\boldsymbol{\theta}}}(x_i)$, due to the coupling of the sub-networks via global network parameters, is weakened when concurrent optimization of multiple sub-networks is replaced by sequential optimization of single sub-networks as is the case for one-sample approximations. The OS-W-dropout variant is therefore not considered in the following. We moreover discard the GL-W-dropout objective (Eq. 5.8) as both the quality of the resulting uncertainty estimates (in preliminary experiments) and its computational "foot print" are similar to standard W-dropout (Eq. 5.4). The subsequent empirical analysis of Wasserstein dropout thus puts a focus on standard W-dropout (Eq. 5.4) and its Gaussian-likelihood one-sample variant (Eq. 5.9).

Comparing the properties of dropout-based uncertainty methods To illustrate qualitative behaviors of different dropout-based uncertainty techniques, namely, Wasserstein dropout, its GL-OS variant and MC dropout, we consider two $\mathbb{R} \to \mathbb{R}$ toy datasets. These benchmarks put emphasis on the handling of heteroscedastic aleatoric uncertainty. The first dataset ("toy-noise") is Gaussian white noise with a x-dependent amplitude, see the blue point cloud in the first row of Fig. 5.1. The second dataset ("toy-hf") is a polynomial overlayed with a high-frequency, amplitude-modulated sine, see the blue curve in the third row of Fig. 5.1. The equations for the toy datasets used here can be found in Appendix B.2.2.

While the uncertainty in the first dataset ("toy-noise") is clearly visible, it is less obvious for the fully deterministic second dataset ("toy-hf"). There is an effective uncertainty though due to the insufficient expressivity of the model, as the shallow networks employed[10] are empirically not able to fit (all) fluctuations of "toy-hf" (see third row of Fig. 5.1). One might (rightfully) argue that this is a sign of insufficient model capacity. For more realistic, e.g., higher dimensional and sparser datasets, however, the distinction between actual noise and complex information becomes exceedingly difficult to make and regularization is actively used to suppress the modeling of stochastic fluctuations. As the Nyquist-Shannon sampling theorem states, with limited data deterministic fluctuations above a cut-off frequency can no longer be resolved (Landau, 1967). They therefore become virtually indistinguishable from random noise.

Training MC dropout- and Wasserstein dropout-based models on these toy datasets and visualizing the sub-network distributions of the optimized models (see Fig. 5.1),

[9] The source code for the GL-OS variant of Wasserstein dropout, which was previously termed *second-moment loss*, can be found at `https://github.com/fraunhofer-iais/second-moment-loss`.

[10] We employ MLPs with two hidden layers each of which contains 50 neurons with ReLU activations.

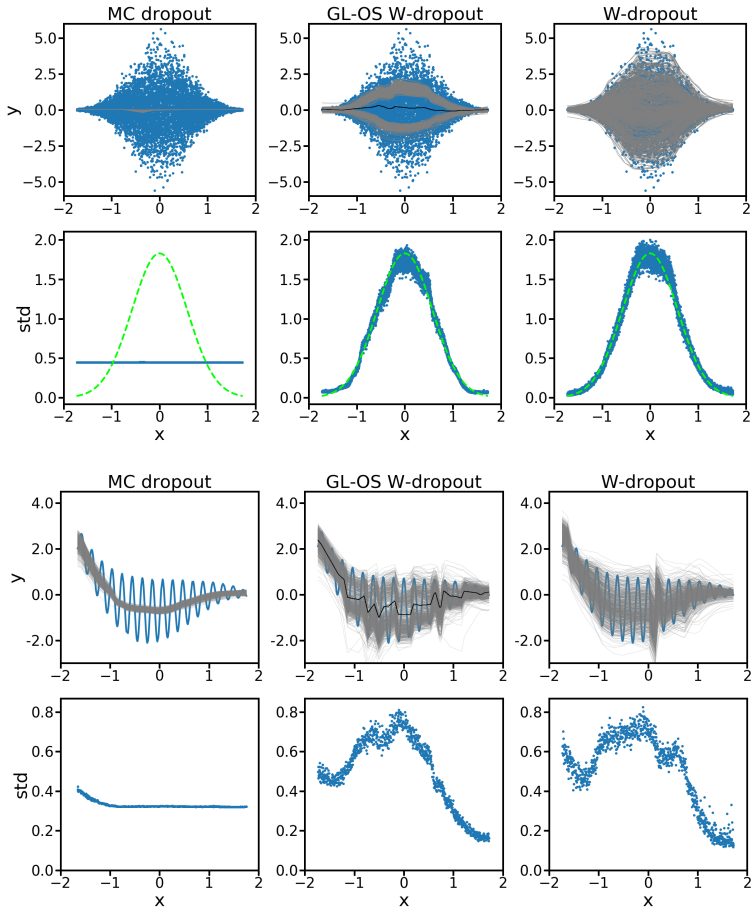


Fig. 5.1.: Comparison of dropout-based uncertainty techniques on toy datasets. MC dropout (left column), the GL-OS variant of Wasserstein dropout (middle column) and standard Wasserstein dropout (right column) are analyzed for a dataset with heteroscedastic noise (blue point cloud in first row) and a high-frequency dataset (blue curve in third row). In particular, their dropout-induced sub-networks are visualized (gray curves in the first and third row) as well as the standard deviations of their output distributions (stds, blue dots in second and fourth row). For the noisy dataset, the ground truth standard deviation is shown for comparison (dashed light green curve in the second row). Focusing on epistemic uncertainty, MC dropout yields (rather) narrow sub-network distributions and fits a constant level of data noise by means of an input-independent parameter (see left panel in second row). The Wasserstein dropout techniques, in contrast, allow for capturing heteroscedastic data uncertainty (second row) and model class uncertainty (fourth row). W-dropout (right column) induces unimodal sub-network distributions whereas its GL-OS variant can bring about bimodality due to one-sample approximations. The deterministic outputs of the GL-OS variant are shown as black curves (first and third row).

the conceptual differences between the different dropout techniques are clearly visible: Wasserstein dropout (Fig. 5.1, right column) and its Gaussian-likelihood one-sample (GL-OS) variant (Fig. 5.1, middle column) employ sub-networks to model *aleatoric uncertainty*. Both input-dependent data noise (of the "toy-noise" dataset, first row) and model class uncertainty (of the "toy-hf" dataset, third row) are reflected in their sub-network distributions. This is in contrast to MC dropout (Fig. 5.1, left column) that uses sub-network distributions to model *epistemic uncertainty*. This type of uncertainty is small after training models on densely sampled toy datasets and consequently MC dropout's sub-network distributions are significantly more narrow compared to the ones of the Wasserstein dropout techniques. MC dropout, however, allows for roughly modeling data-intrinsic uncertainty by means of a tunable "offset" hyperparameter that is chosen to match the average noise level of a dataset (see the left panels in the second and fourth row of Fig. 5.1).

Differences between W-dropout and its GL-OS variant manifest in the properties of the output distributions they induce: standard W-dropout yields unimodal distributions whereas the GL-OS variant can bring about bimodality. This behavior of GL-OS is due to the structure of the second summand of its optimization objective (Eq. 5.9) that removes (as stated above) the directional information of the residuals of the deterministic network. For 1D datasets, a sub-network output may lie, figuratively speaking, "above" or "below" the output of the deterministic network (that is visualized as a black curve in Fig. 5.1). Changes between these "orientations" as a function of x rarely occur for the datasets in Fig. 5.1 as such "switches" imply an increase of the second loss term and additional complexity of the respective sub-network. An (approximate) analytical treatment of the loss landscape of the GL-OS variant as well as further analyses of its loss components can be found in Appendix B.1. While the GL-OS variant of Wasserstein dropout provides competitive uncertainty quality, it comes along with output bimodality that does not reflect any actual (stochastic) input-output relation in the dataset to be modeled. It is a modeling artifact, which may, moreover, negatively impact downstream tasks that build on distributional properties beyond the second central moment. In the following empirical studies, we thus discard the GL-OS loss variant and focus on the standard Wasserstein dropout technique.

5.2. Uncertainty assessment beyond standard measures

To compare the output distributions induced by Wasserstein dropout more systematically with those of MC dropout and other uncertainty estimation techniques (see Section 5.3), measures for similarities and deviations between probability distributions are required. These scores of uncertainty quality (see Section 3.3 for an overview and the next paragraph for a concise recapitulation) are typically scalar, i.e., they map (potentially complex) distributional differences onto single values. Being aggregating in that sense, they necessarily put emphasis on some distributional properties while neglecting others. Here, we propose two novel uncertainty scores that address shortcomings of established uncertainty

measures. We argue that the proposed uncertainty scores complement the existing ones and thus help to "draw" a more comprehensive picture of uncertainty quality. In particular, an unbounded calibration measure is introduced (second paragraph) as well as an uncertainty tail measure for the analysis of worst-case scenarios w.r.t. uncertainty quality (third paragraph).

Standard evaluation measures In the following experiments, we evaluate both regression performance and uncertainty quality. Regression performance is quantified by the root-mean-square error, $\sqrt{1/N \sum_i (\mu_i - y_i)^2}$ (**RMSE**, see Section 2.3). We moreover consider the (Gaussian) negative log-likelihood (**NLL**), $1/N \sum_i \left(\log \sigma_i + (\mu_i - y_i)^2/(2\sigma_i^2) + c \right)$, a hybrid between performance and uncertainty measure (see Section 2.3 and, for further discussion, Appendix B.4.2). Throughout this chapter, we ignore the constant $c = \log \sqrt{2\pi}$ of the NLL. The expected calibration error (**ECE**, see Section 2.3) in contrast is not biased toward well-performing models and in that sense a pure uncertainty measure. Technically, it is based on so-called *normalized prediction residuals* r_i that are defined as $r_i = (\mu_i - y_i)/\sigma_i$. Assuming B equally spaced bins in quantile space, the ECE reads

$$\mathrm{ECE}(\{r_i\}) = \sum_{j=1}^{B} \left| \tilde{p}_j(\{r_i\}) - \frac{1}{B} \right| \ , \tag{5.11}$$

where $\tilde{p}_j(\{r_i\}) = |\{r_i | q_j \leq \tilde{q}(r_i) < q_{j+1}\}|/N$ denotes the empirical frequency of data points falling into the bin $[q_j, q_{j+1})$ and $\tilde{q}$ the cumulative distribution function (CDF) of the standard normal distribution $\mathcal{N}(0, 1)$.

An unbounded uncertainty calibration measure A desirable property for uncertainty measures is a signal that grows (preferentially linearly) with the misalignment between predicted and ideal uncertainty estimates, especially when handling strongly deviating uncertainty estimates. As the Wasserstein metric fulfills this property, we not only use it for model optimization but propose to consider the *1-Wasserstein distance of normalized prediction residuals* (**WS**) as a complementary uncertainty evaluation measure. It is generally applicable and by no means restricted to W-dropout networks. In detail, the 1-Wasserstein distance (Villani, 2008), also known as earth mover's distance (Rubner et al., 1998), is a transport-based measure, denoted by d_{WS}, between two probability densities, with Wasserstein GANs (Arjovsky et al., 2017) as its most prominent application in machine learning. In the context of uncertainty estimation, we use the Wasserstein distance to measure deviations of uncertainty estimates $\{r_i\}_i$ from ideal (Gaussian)[11] calibration that is given if $y_i \sim \mathcal{N}(\mu_i, \sigma_i)$ with accompanying normalized residuals of $r_i \sim \mathcal{N}(0, 1)$, i.e., we calculate $d_{\mathrm{WS}}\left(\{r_i\}_i, \mathcal{N}(0, 1)\right)$. As ECE, this is a pure uncertainty measure. However, it is not based on quantiles but directly on normalized residuals and can therefore resolve deviations on all scales. For example, two strongly ill-calibrated

[11] As stated before, Gaussianity, while not always given, is a standard assumption for uncertainty modeling and typically used in ECE and NLL.

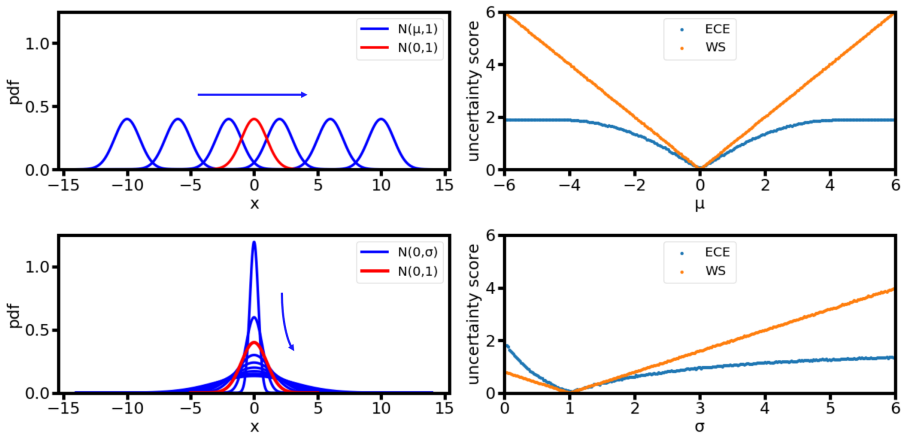


Fig. 5.2.: Comparison of the proposed Wasserstein-based measure (WS) and the expected calibration error (ECE). We measure the deviation between a standard normal distribution $\mathcal{N}(0,1)$ (l.h.s., red) and shifted normal distributions $\mathcal{N}(\mu,1)$ (top left, dark blue) as well as the deviation between $\mathcal{N}(0,1)$ and squeezed or stretched normal distributions $\mathcal{N}(0,\sigma)$ (bottom left, dark blue). The resulting ECE values (orange) and WS values (blue) on the r.h.s. emphasize the higher sensitivity of WS in the case of large distributional differences. For details on ECE and WS, see the text in Section 5.2.

uncertainties would result in (almost) identical ECE values while WS would resolve this difference in magnitude. Let us compare ECE and WS more systematically: we consider normal distributions $\mathcal{N}(\mu,1)$ and $\mathcal{N}(0,\sigma)$ (see Fig. 5.2) that are shifted (top left panel, dark blue) or deformed by squeezing or stretching (bottom left panel, dark blue). Their deviations from the ideal normalized residual distribution (the standard normal, red) are measured in terms of both ECE (r.h.s., blue) and WS (r.h.s., orange). For large values of $|\mu|$ and σ, ECE is bounded while WS increases linearly showing the better sensitivity of the latter toward strong deviations. For small values, $\sigma \to 0$, ECE takes its maximum value, WS a value of 1. In Fig. 5.3, we visualize these value pairs (WS(σ), ECE(σ)) (gray lines), i.e., σ serves as curve parameter. The upper branch corresponds to $0 < \sigma < 1$, the lower branch to $\sigma > 1$. For comparison, the pairs (WS, ECE) of various networks trained on standard regression datasets are visualized (see Subsection 5.3.3 for experimental details and results). They approximately follow the theoretical σ-curve, emphasizing that both under- and overestimated variances are of practical relevance. A given WS value allows, due to lacking saturation for underestimation, to distinguish these two cases more easily compared to ECE. While one might rightfully argue that the higher sensitivity of WS leads to a certain susceptibility to potential outliers, this can be addressed by regularizing the normalized residuals or by filtering extreme outliers.

A novel uncertainty tail measure We furthermore introduce a measure for distributional tails that allows us to analyze worst-case scenarios w.r.t. uncertainty quality, thus reflecting

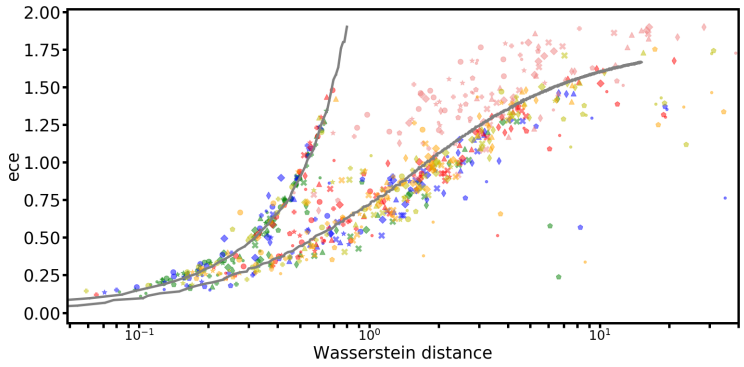


Fig. 5.3.: Dependency between the Wasserstein-based measure and the expected calibration error for Gaussian toy data (gray curves) and for 1D standard datasets (point cloud, see Subsection 5.3.3 for details). The toy curves are obtained by plotting (WS(σ), ECE(σ)) from Fig. 5.2 (bottom right). For 1D standard datasets, uncertainty methods are encoded via plot markers, data splits via color. Datasets are not encoded and cannot be distinguished (see Appendix B.4 for more details). Each plot point corresponds to a cross-validated trained network.

safety considerations. Such potentially critical worst-case scenarios are signified by the above-mentioned outliers, where the locally predicted uncertainty strongly underestimates the actual model error. A better understanding of uncertainty estimates in these scenarios can help to determine lower bounds on operation quality of safety-critical systems. For this, we consider normalized residuals $r_i = (\mu_i - y_i)/\sigma_i$ based on the prediction estimates (μ_i, σ_i) for a given data point (x_i, y_i). As stated, we restrict our analysis to uncertainty estimates that *underestimate* model errors, i.e., $|r_i| \gg 1$. These cases might be more harmful than overly large uncertainties, $|r_i| \ll 1$, that likely trigger a conservative system behavior. We quantify uncertainty quality for worst-case scenarios as follows: for a given (test) dataset, the absolute normalized residuals $\{|r_i|\}_i$ are calculated. We determine the 99% quantile $q_{0.99}$ of this set and calculate the mean value over all $|r_i| > q_{0.99}$, the so-called *expected tail loss at quantile* 99% ($\text{ETL}_{\mathbf{0.99}}$, Rockafellar & Uryasev (2002)). The $\text{ETL}_{0.99}$ thus measures the average uncertainty quality of the worst 1%.

5.3. Empirical evaluation on 1D regression datasets

We first outline the scope of our empirical study in Subsection 5.3.1 and revisit the illustrative and visualizable toy datasets that were used above in Subsection 5.3.2. Finally, we benchmark Wasserstein dropout on various 1D datasets (mostly from the UCI machine learning repository (Dua & Graff, 2017)) in Subsection 5.3.3, considering both in-data test sets and distribution-shift scenarios.

5.3.1. Experiment setup

In this subsection, we present the considered benchmark approaches (first paragraph) and give a brief overview of the employed technical setup (second paragraph).

Benchmark approaches We compare W-dropout networks to archetypes of uncertainty modeling, namely, approximate Bayesian techniques, parametric uncertainty, and ensembling approaches. From the first group, we pick MC dropout (abbreviated as **MC**, Gal & Ghahramani (2016a)) and Concrete dropout (**CON-MC**, Gal et al. (2017)). The variance of MC is given as the sample variance plus a dataset-specific regularization term. The networks employing these methods do not exhibit parametric uncertainty outputs (see below). We additionally consider SWA-Gaussian (**SWAG**, Maddox et al. (2019)), which samples from a Gaussian model weight distribution that is constructed based on model parameter configurations along the (final segment of the) training trajectory. While these sampling-based approaches integrate uncertainty estimation into the structure of the entire network, *parametric* approaches model the variance directly as the output of the neural network (Nix & Weigend, 1994). Such networks typically output mean and variance of a Gaussian distribution (μ, σ^2) and are trained by likelihood maximization. This approach is denoted as **PU** for parametric uncertainty. Ensembles of PU-networks (Lakshminarayanan et al., 2017), referred to as deep ensembles, pose a widely used state-of-the-art method for uncertainty estimation (Snoek et al., 2019). Deep evidential regression (**PU-EV**, Amini et al. (2020)) extends this parametric approach and considers prior distributions over μ and σ. Kendall & Gal (2017) consider drawing multiple dropout samples from a parametric uncertainty model and aggregating multiple predictions for μ and σ. We denote this approach as **PU-MC**. Moreover, we consider ensembles of non-parametric standard networks. We refer to the latter ones as **DEs** while we call those using additionally PU-based uncertainty **PU-DEs**. All considered types of networks provide estimates (μ_i, σ_i) where σ_i is obtained either as direct network output (PU, PU-EV), by sampling (MC, CON-MC, SWAG, W-dropout) or as an ensemble aggregate (DE, PU-DE). For PU-MC, a combination of parametric output and sampling is employed. Throughout this section, we subsume PU, PU-EV, PU-DE and PU-MC as *parametric methods*.

Technical setup Throughout this subsection, we use almost identical networks with two hidden layers and ReLu activations, employing 50 neurons per layer for the toy datasets and 100 for the 1D standard datasets. All dropout-based networks (MC, CON-MC, W-dropout) apply Bernoulli dropout to all hidden activations. For W-dropout networks, we sample $L = 5$ sub-networks in each optimization step, other values of L are considered in Appendix B.3. On the smaller toy datasets, we afford $L = 10$. For MC and W-dropout, the drop rate is set to $p = 0.1$ (see Appendix B.3 for other values of p). The drop rate of CON-MC in contrast is learned during training and (mostly) takes values between $p = 0.2$ and $p = 0.5$. For ensemble methods (DE, PU-DE) we employ five networks. All NNs are optimized using the Adam optimizer (Kingma & Ba, 2015) with a learning rate of $\mu = 0.001$. Additionally, we apply standard normalization to the input and output

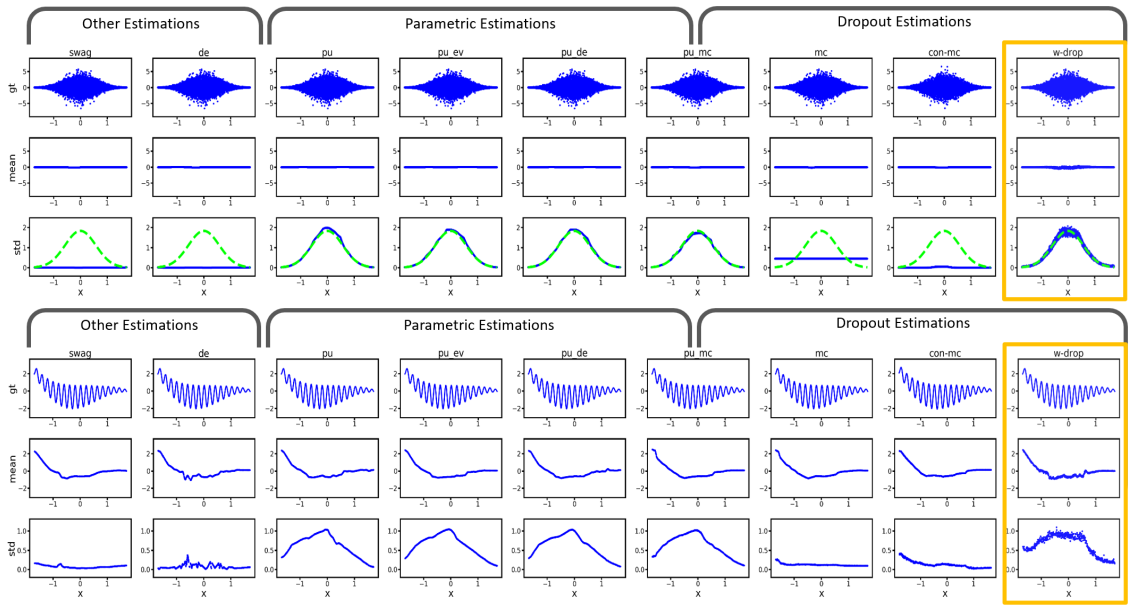


Fig. 5.4.: Comparison of uncertainty approaches (columns) on two 1D toy datasets: a noisy one (top) and a high-frequency one (bottom). Test data ground truth (respective first row) is shown with mean estimates (resp. second row) and standard deviations (resp. third row). The light green dashed curve (third row) indicates the ground truth uncertainty. Similar uncertainty approaches (columns) are grouped together, W-dropout is highlighted by a yellow frame.

features of all datasets to enable better comparability. The number of training epochs and cross validation runs depends on the dataset size. Further technical details on the networks, the training procedure, and the implementation of the uncertainty methods can be found in Appendix B.2.1. In using a least squares regression, we make the standard assumption that errors follow a Gaussian distribution. This is reflected in the (standard) definitions of above named measures, i.e., all uncertainty measures quantify the set of outputs $\{(\mu_i, \sigma_i)\}$ relative to a Gaussian distribution.

5.3.2. Toy datasets

In this subsection, we revisit the two 1D toy datasets from Section 5.1, namely, the heteroscedastic noise dataset ("toy-noise") and the high-frequency dataset ("toy-hf"). They were employed in Section 5.1 to compare Wasserstein dropout (and a variant of it) with MC dropout. Here, we extend this qualitative analysis to all uncertainty estimation techniques outlined in Subsection 5.3.1.

Figure 5.4 shows, similar to Fig. 5.1, the "toy-noise" and the "toy-hf" datasets (first and fourth row) and the output standard deviations induced by the considered benchmark methods (third and sixth row). Moreover, the mean predictions of the uncertainty methods are displayed (second and fifth row). On both datasets, the respective mean estimates

look almost identical for all uncertainty methods. They approximate the mean value of the noise and the polynomial, respectively. For "toy-hf", some methods rudimentarily fit individual fluctuations. The variance estimation (third and sixth row in Fig. 5.4), in contrast, reveals significant differences between the methods: MC dropout variants and other non-parametric ensembles are not capable of capturing heteroscedastic aleatoric uncertainty. This behavior of MC is expectable as it was primarily introduced to account for model parameter uncertainty. The non-parametric DE is effectively optimized in a similar fashion. In contrast, NLL-optimized PU networks have a home-turf advantage on these datasets since the parametric variance is explicitly optimized to account for the present heteroscedastic aleatoric uncertainty. W-dropout is the only non-parametric approach that accounts for the presence of this kind of uncertainty. While the results look similar, the underlying mechanisms are fundamentally different. On the one hand explicit prediction of the uncertainty, on the other hand implicit modeling via distribution matching. Accompanying quantitative evaluations can be found in Appendix B.2.2. To collect further evidence that W-dropout approximates the ground truth uncertainty σ_{true} appropriately, we fit it to "noisy-line" toy datasets in Appendix B.2.2. Both large and small σ_{true} values are correctly matched, indicating that W-dropout is not just adding an uncertainty offset but flexibly spreads and contracts its sub-networks as intended. In the following, we substantiate the corroborative results of W-dropout on toy data by an empirical study on 1D standard datasets and an application to a modern object detection network.

5.3.3. Standard ML datasets

Next, we study standard regression datasets, extending the dataset selection in Gal & Ghahramani (2016a) by adding four additional datasets: "diabetes", "abalone", "california" and "superconduct". Tab. B.2 (p. 156) in Appendix B.2.3 provides details on dataset sources, preprocessing and basic statistics. Apart from train- and test-data results, we study regression performance and uncertainty quality *under data shift*. Such distributional changes and uncertainty quantification are closely linked since the latter ones are rudimentary self-assessment mechanisms that help to judge model reliability. These judgments gain importance for model inputs that are *structurally different* from the training data.

Data splits Natural candidates for such non-i.i.d. splits are splits along the main directions of data in input and output space, respectively. Here, we consider 1D regression tasks. Therefore, output-based splits are simply done on a scalar label variable (see Fig. 5.5, right). We call such a split *label-based* (for a comparable split, see, e.g., Foong et al. (2019)). In input space, the first component of a principal component analysis (PCA) provides a natural direction (see Fig. 5.5, left). Projecting the data points onto this first PCA-axis yields the scalar values the *PCA-split* is based on. Note that these projections are only considered for data splitting, they are not used for model training. Splitting data along such a direction in input or output space in, e.g., ten equally large chunks, creates two *outer* data chunks and eight *inner* data chunks. Training a model on nine of

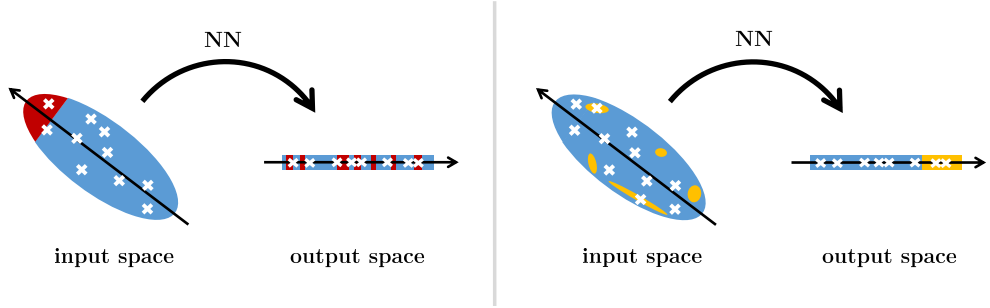


Fig. 5.5.: Scheme of two non-i.i.d. splits: a PCA-based split in input space (left) and label-based split in output space (right). While datasets appear to be convex here, they are (most likely) not in reality.

these chunks such that the remaining chunk for evaluation is an inner chunk is called data *interpolation*. If the remaining test chunk is an outer chunk, it is data *extrapolation*. For example, for labels running from 0 to 1, (label-based) extrapolation testing could consider only data with a label larger 0.9, while training would be performed on the smaller label values. We introduce this distinction as extrapolation is expected to be considerably more difficult than bridging between feature combinations that were seen during training.

More general information on training and dataset-dependent modifications to the experimental setup are relegated to the technical Appendix B.2.1. The presented results are obtained as follows: for each of the 14 standard datasets, we calculate (for each uncertainty method) the per-dataset scores: RMSE, mean NLL, ECE and WS. To improve statistical significance, these scores are 5- or 10-fold cross-validated, i.e., averages across a respective number of folds. Given the (fold-averaged) per-dataset scores for all 14 standard datasets, we calculate and visualize their mean and median values as well as quantile intervals (see Fig. 5.6 and Fig. 5.7). For high-level summaries of the results on in-data and out-of-data test sets, refer to Tab. 5.1 and Tab. 5.2, respectively. While the mean values characterize the average behavior of the uncertainty methods, the displayed 75% quantiles indicate how well methods perform on the more challenging datasets. A small 75% quantile value thus hints at *consistent* stability of an uncertainty mechanism across a variety of tasks.

Regression quality First, we consider regression performance, see Tab. 5.1 and the first two panels in the top row of Fig. 5.6. Averaging the RMSE values across the 14 datasets yields almost identical test results for all uncertainty methods (see Tab. 5.1). On training data (Fig. 5.6, first panel in top row) in contrast, we find the parametric methods to exhibit larger train data RMSEs, which could be due to NLL optimization favoring to adapt variance rather than mean. However, this regularizing NLL training comes along with a smaller generalization gap, leading to competitive test RMSEs (see Tab. 5.1 and the second panel in the top row of Fig. 5.6). W-dropout is on a par with the benchmark approaches, i.e., our optimization objective does not lead to degraded

Tab. 5.1.: Regression performance (RMSE) and uncertainty quality (NLL, ECE, WS) of W-dropout and various uncertainty benchmarks. W-dropout yields the best uncertainty scores while providing a competitive RMSE value. Each number is the average across 14 standard 1D (test) datasets. The figures in this table correspond to the blue crosses in the second columns of Fig. 5.6 and Fig. 5.7, respectively. See the text in Subsection 5.3.3 for further details.

unc. method	**RMSE** (↓)	**NLL** (↓)	**ECE** (↓)	**WS** (↓)
SWAG	0.456	7.695	0.828	1.847
DE	**0.407**	6.184	0.796	1.628
PU	0.447	$1.47 \cdot 10^7$	0.614	$2.10 \cdot 10^6$
PU-EV	0.442	2.838	0.626	49.180
PU-DE	0.418	0.307	0.515	0.542
PU-MC	0.420	0.250	0.565	0.433
MC	0.412	0.190	0.788	0.643
CON-MC	0.436	1.513	0.669	0.964
W-Dropout	0.421	**−0.428**	**0.501**	**0.430**

regression quality.[12] Next, we investigate model performance under data shift, visualized in the third to sixth panel in the top row of Fig. 5.6. For interpolation setups (fourth and sixth panel), regression quality is comparable between all methods. As expected, performances under these data shifts are (slightly) worse compared to those on i.i.d. test sets. The more challenging extrapolation setups (third and fifth panel) amplify the deterioration in performance across all methods. Again, W-dropout yields competitive RMSE values (see also Tab. 5.2).

Expected calibration errors Fig. 5.6 (bottom row) provides average ECE values of the outlined uncertainty methods under i.i.d. conditions (first and second panel), under label-based data shifts (third and fourth panel) and under PCA-based data shifts (fifth and sixth panel). On training data, PU performs best, followed by PU-EV and all other methods. Interestingly, both SWAG and W-dropout show a relatively broad range of ECE values on the various training datasets. This could be interpreted as a form of over-estimation of the present uncertainty and for W-dropout this effect occurs on mostly smaller datasets with lower data variability. However, looking at the i.i.d. test results (Tab. 5.1 and second panel in the bottom row of Fig. 5.6) we find W-dropout to provide the lowest averaged ECE (Tab. 5.1), followed by the PU-based (implicit) ensembles of PU-DE and PU-MC. The calibration quality of W-dropout is moreover the most consistent one across the datasets as can be seen from its small 75% quantile value (Fig. 5.6, second panel in bottom row).

[12] This observation does not only hold relative to the other uncertainty methods but, moreover, relative to a deterministic network, see Fig. B.6 (p. 158) and surrounding discussions in Appendix B.2.3.

Tab. 5.2.: Out-of-data analysis of W-dropout and various uncertainty benchmarks. Regression performance (RMSE) and uncertainty quality (NLL, ECE, WS) are displayed. As for in-domain test data, W-dropout outperforms the other uncertainty methods without sacrificing regression quality. Each number is obtained by two-fold averaging: first, across two types of out-of-data test sets (label-based and PCA-based splits) and second, across 14 standard 1D datasets. The figures in this table are based on the blue crosses in the last four columns of Fig. 5.6 and Fig. 5.7, respectively. See the text in Subsection 5.3.3 for further details.

unc. method	**RMSE** (↓)	**NLL** (↓)	**ECE** (↓)	**WS** (↓)
SWAG	0.641	27.602	1.138	3.818
DE	0.599	14.055	0.988	2.554
PU	0.632	$1.50 \cdot 10^7$	0.968	$1.55 \cdot 10^6$
PU-EV	0.611	6.290	0.941	44.447
PU-DE	0.594	1448.391	0.783	5.892
PU-MC	0.591	397.022	0.823	3.215
MC	**0.589**	2.330	0.923	1.207
CON-MC	0.621	13.820	0.963	2.109
W-Dropout	0.615	**2.287**	**0.763**	**1.203**

Looking at the stability w.r.t. data shift, i.e., extra- and interpolation based on label-split or PCA-split, again W-dropout reaches the smallest calibration errors (followed by PU-DE and PU-MC, see Tab. 5.2). Regarding the 75% quantiles, W-dropout consistently provides one of the best results on all out-of-data (OOD) test sets.

Negative log-likelihoods For the unbounded NLL (see Tab. 5.1 and the top row of Fig. 5.7), the results are more widely distributed compared to the (bounded) ECE values. W-dropout reaches the smallest mean value on i.i.d. test sets, followed by MC and PU-MC (Tab. 5.1). The mean NLL value of PU is above the upper plot limit in Fig. 5.7 (second panel in the upper row) indicating a rather weak stability of this method. On PCA-interpolate and PCA-extrapolate test sets (Fig. 5.7, last two panels in the upper row), MC, PU-MC and W-dropout networks perform best. On label-interpolate and label-extrapolate test sets, only MC and W-dropout networks are in first place when considering average values, followed by PU-EV. The mean NLLs of many other approaches are above the upper plot limit. Averaging all these OOD results in Tab. 5.2, we find W-dropout to provide the overall smallest NLL values, narrowly followed by MC. Note that median results are not as widely spread and PU-DE, MC, PU-MC and W-dropout perform comparably well. These qualitative differences between mean and median behavior indicate that most methods perform poorly "once in a while". A noteworthy observation as *stability across a variety of data shifts and datasets* can be seen as a crucial requirement for an uncertainty method. W-dropout models yield high stability in that sense w.r.t. NLL.

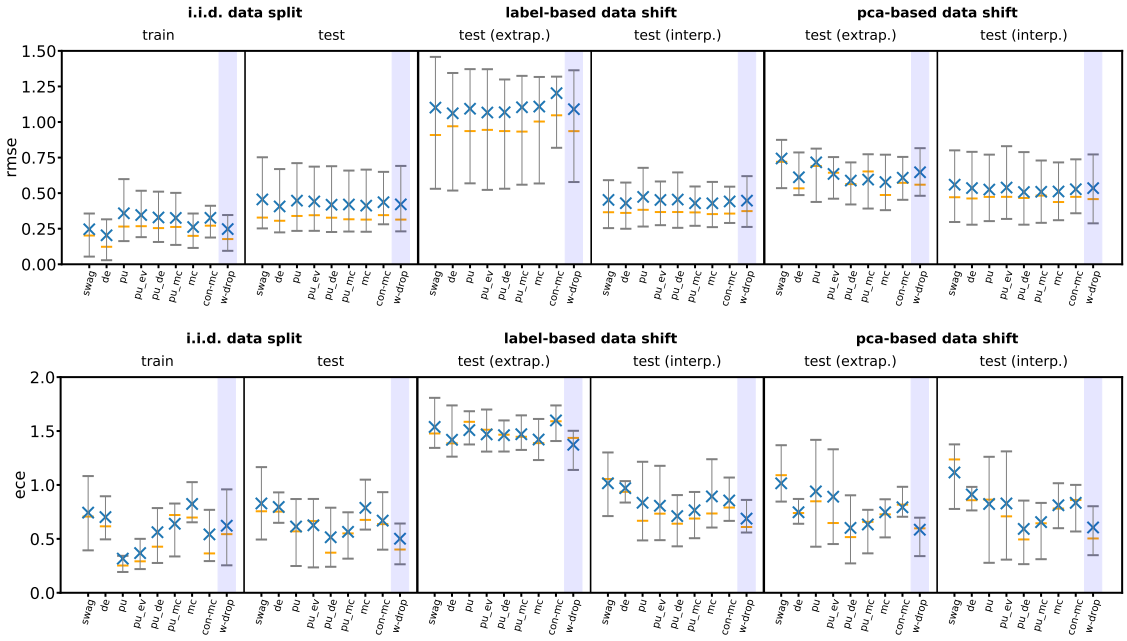


Fig. 5.6.: Root-mean-square errors (RMSEs (↓), top row) and expected calibration errors (ECEs (↓), bottom row) of different uncertainty methods under i.i.d. conditions (first and second panel in each row) and under various kinds of data shift (third to sixth panel in each row, see the text in Subsection 5.3.3 for details). W-dropout (light blue background) is compared to 8 benchmark approaches. Each blue cross is the mean over 14 1D regression datasets. Orange line markers indicate median values. The gray vertical bars reach from the 25% quantile (bottom horizontal line) to the 75% quantile (top horizontal line).

Wasserstein distances Studying Wasserstein distances, we again observe the smallest scores on test data for W-dropout, followed by PU-MC and PU-DE (see Tab. 5.1 and the second panel in the bottom row of Fig. 5.7). While PU provides the best WS value on training data, its generalization behavior is less stable: on test data, its mean and 75% quantile take high values beyond the plot range. Under data shift (Tab. 5.2 and third to sixth panel in bottom row of Fig. 5.7), W-dropout and MC are in the lead, CON-MC and DE follow on ranks three and four. On label-based data shifts, MC and W-dropout outperform all other methods by a significant margin when considering average values. As for NLL, we find the mean values for PU-DE and PU-MC to be significantly above their respective median values indicating again weaknesses w.r.t. the stability of parametric methods. Here as well, not only good *average* results, but also consistency over the datasets and splits, is a hallmark of Wasserstein dropout.

Epistemic uncertainty Summarizing these evaluations on 1D regression datasets, we find W-dropout to yield better and more stable uncertainty estimates than the state-of-the-art methods of PU-DE and PU-MC. We moreover observe advantages for W-dropout

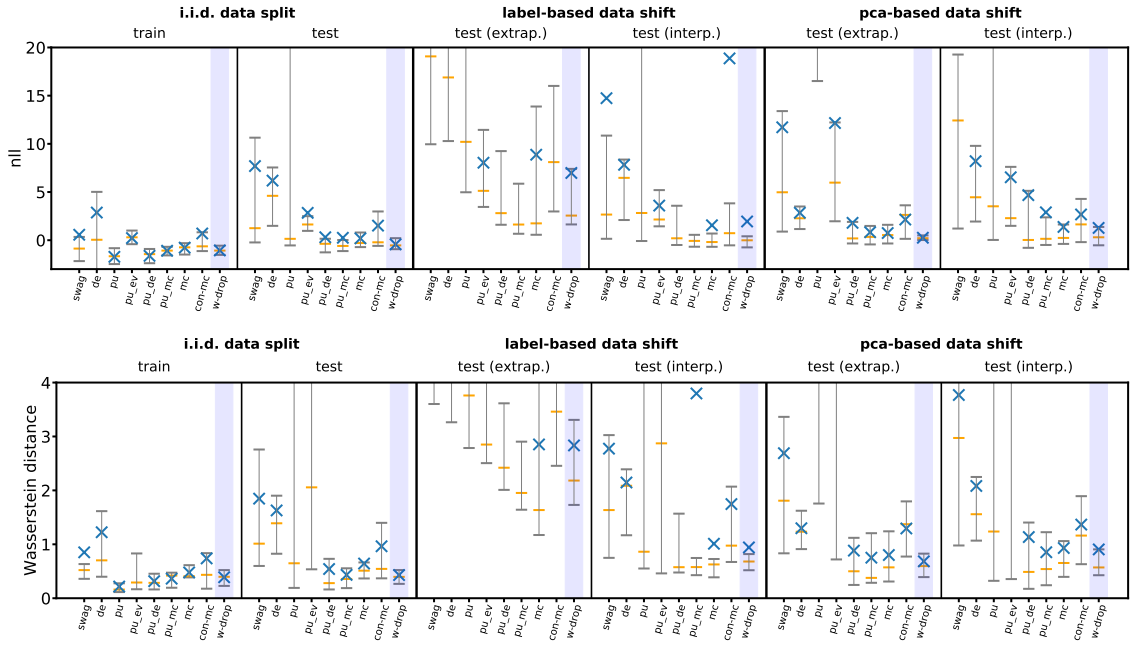


Fig. 5.7.: Negative log-likelihoods (NLLs (↓), top row) and Wasserstein distances (↓, bottom row) of different uncertainty methods under i.i.d. conditions (first and second panel in each row) and under various kinds of data shift (third to sixth panel in each row, see the text in Subsection 5.3.3 for details). W-dropout (light blue background) is compared to 8 benchmark approaches. Each blue cross is the mean over ECE values from 14 standard regression datasets. Orange line markers indicate median values. The gray vertical bars reach from the 25% quantile (bottom horizontal line) to the 75% quantile (top horizontal line).

under PCA- and label-based data shifts. These results suggest that W-dropout induces uncertainties that increase under data shift, i.e., it approximately models *epistemic uncertainty*. This conjecture is supported by Fig. 5.8 that visualizes the uncertainties of MC dropout (blue) and W-dropout (orange) for transitions from in-data to out-of-data. As expected, these shifts lead to increased (epistemic) uncertainty for MC dropout. This holds true for W-dropout that behaves highly similar under data shift indicating that it "inherits" this ability from MC dropout: both approaches match sub-networks to training data and these sub-networks spread when leaving the training data distribution. Since W-dropout models heteroscedastic, i.e., input-dependent, aleatoric uncertainty, we notice a higher variability of its uncertainties in Fig. 5.8 compared to the ones of MC dropout.

For further (visual) inspections of uncertainty quality, see the residual-uncertainty scatter plots in Appendix B.2.4. A reflection on NLL and comparisons of the different uncertainty measures on 1D regression datasets can be found in Appendix B.2.3.

Expected tail loss For both toy and standard regression datasets, we calculate the expected tail loss at the 99% quantile ($\text{ETL}_{0.99}$) on test data. Doing this for all trained

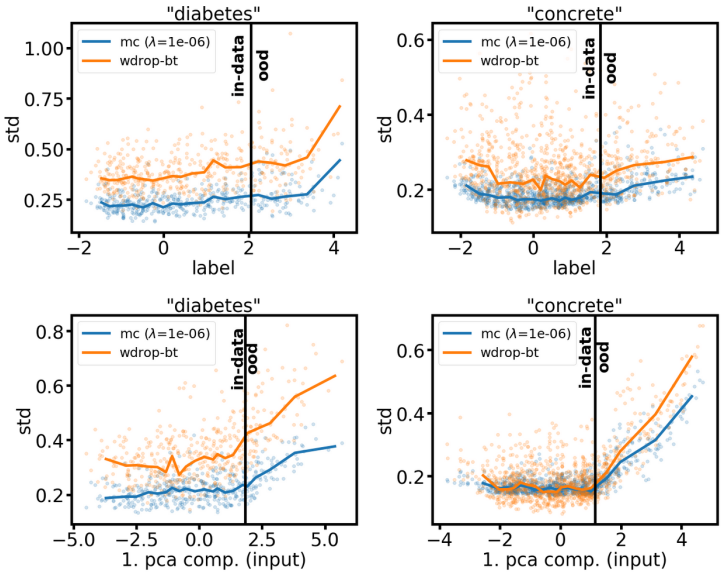


Fig. 5.8.: Extrapolation behavior of W-dropout (orange) and MC dropout (blue). Two extrapolation "directions" (rows) and two datasets (columns) are considered. The vertical bar in each panel separates training data (left) from out-of-data (OOD, right). Scatter points show the predicted standard deviation for individual data points. The colored solid lines show averages over points in equally-sized bins and reflect the expected growth of epistemic uncertainty in the OOD-region. For details on the data splits and extrapolations, refer to Subsection 5.3.3 and Appendix B.2.3.

networks yields a total of 110 $\text{ETL}_{0.99}$ values per uncertainty method when including cross-validation. As a tail measure, the $\text{ETL}_{0.99}$ evaluates a specific aspect of the distribution of uncertainty estimates. Studying such a property is useful if the uncertainty estimate distribution as a whole is appropriate, as measured, e.g., by the ECE. We thus restrict the $\text{ETL}_{0.99}$ analysis to the three methods that provide the best ECE values, namely, PU-MC, PU-DE and W-dropout. The mean and maximum values of their $\text{ETL}_{0.99}$'s are reported in Tab. 5.3. While none of these methods gets close to the ideal $\text{ETL}_{0.99}$'s of the desired $\mathcal{N}(0,1)$ Gaussian, W-dropout networks exhibit significantly less pronounced tails and therefore higher stability compared to PU-MC and PU-DE. This holds true over all considered test sets. Deviations from standard normal increase from the i.i.d. train-test split over the PCA-based train-test split to the label-based one. We attribute the lower stability of PU-DE to the nature of the PU networks that compose the ensemble, although their inherent instability (see Tab. B.3 (p. 157) in Appendix B.2.3) is largely suppressed by ensembling. Considering the tail of the distribution of the prediction residuals $|r_i|$, however, reveals that regularization of PU by ensembling might not work in every single case. It is then unlikely that larger ensemble are able to fully cure this instability issue. Regularizing PU by applying dropout (PU-MC) leads to only mild improvement. W-dropout networks in contrast encode uncertainty into the structure of

Tab. 5.3.: Study of worst-case scenarios for different uncertainty methods: W-dropout (W-Drop), PU-DE and PU-MC are compared to the ideal Gaussian case for i.i.d. and non-i.i.d. data splits. Uncertainty quality in these scenarios is quantified by the expected tail loss at the 99% quantile ($\mathrm{ETL}_{0.99}$). Each mean and max value is taken over the ETLs of 110 models trained on 15 different datasets.

measure	split	$\mathcal{N}(\mathbf{0},\mathbf{1})$	W-Drop	PU-DE	PU-MC
mean $\mathrm{ETL}_{0.99}$	i.i.d.	2.89	4.68	7.77	6.24
max $\mathrm{ETL}_{0.99}$	i.i.d.	3.01	8.86	31.14	91.75
mean $\mathrm{ETL}_{0.99}$	label	2.89	7.28	86.79	44.00
max $\mathrm{ETL}_{0.99}$	label	3.01	93.55	2267.93	1224.27
mean $\mathrm{ETL}_{0.99}$	pca	2.89	5.93	9.78	8.62
max $\mathrm{ETL}_{0.99}$	pca	3.01	18.35	64.13	93.49

the entire network thus yielding improved stability compared to parametric approaches. Further analysis shows that the large normalized residuals $r_i = (\mu_i - y_i)/\sigma_i$, which cause the large $\mathrm{ETL}_{0.99}$ values, correspond (on average) to large absolute errors $(\mu_i - y_i)$.[13] This underpins the practical relevance of the ETL analysis, as large absolute errors are more harmful than small ones in many contexts, e.g., when detecting traffic participants.

Dependencies between uncertainty measures All uncertainty-related measures (NLL, ECE, WS, ETL) relate predicted uncertainties to actually occurring model residuals. Each of them putting emphasize on different aspects of the considered samples: NLL is biased toward well-performing models, ECE measures deviations within quantile ranges, Wasserstein distance resolves distances between normalized residuals and ETL focuses on distribution tails. The empirically observed dependencies between WS and ECE are visualized in Fig. 5.3. Additionally to WS and ECE, we consider Kolmogorov-Smirnov (KS) distances (Stephens, 1974) on normalized residuals in Fig. B.15 (p. 171) in Appendix B.4.

While all these scores are expectably correlated, noteworthy deviations from ideal correlation occur. Therefore, we advocate for uncertainty evaluations based on various measures to avoid overfitting to a specific formalization of uncertainty. The top panel of Fig. B.15 reflects the higher sensitivity of the Wasserstein distance compared to ECE: we observe two slopes, the first one corresponds to models that overestimate uncertainties, i.e., $\sigma_{\tilde{\boldsymbol{\theta}}} > |\mu_{\tilde{\boldsymbol{\theta}}} - y_i|$ on average. In these scenarios, WS is typically below 1 as 1 would be the WS distance between a delta distribution at zero (corresponding to $\sigma_{\tilde{\boldsymbol{\theta}}} \to \infty$) and the expected $\mathcal{N}(0,1)$ Gaussian. The second slope contains models that underestimate

[13] They are (on average) not due to small absolute residuals $\ll 1$ that go along with even smaller uncertainty estimates.

uncertainties, i.e., $\sigma_{\tilde{\boldsymbol{\theta}}} < |\mu_{\tilde{\boldsymbol{\theta}}} - y_i|$. WS is not bounded in these scenarios and is thus—unlike ECE—able to resolve differences between any two uncertainty estimators.

5.4. Wasserstein dropout for object detection

After studying toy and standard regression datasets, we turn toward the challenging task of object detection (OD, Zhao et al. (2019); Liu et al. (2020)), i.e., the localization and classification of instances of semantic classes in unstructured data such as images or LiDAR point clouds. One may, for instance, think of the detection of traffic participants like pedestrians, cyclists and vehicles on camera recordings.[14] These extractions of high-level semantic information from low-level data render OD a central pillar for the automated perception of environments, which is in turn the basis for more complex ML systems like autonomous agents, e.g., in the field of mobility. On a technical level, OD models can be characterized by the type of input data they process (for instance, monocular or binocular camera images, single frames or video sequences, one or multiple types of sensors) and the kind of object information they predict (for instance, 2D or 3D object coordinates, one or multiple object classes). Here, we employ (in these regards rather basic) OD models that predict 2D object coordinates and object classes from single images. However, compared to the experiments above on 1D datasets, even this basic OD model setup comes along with several additional complexities that range from high input dimensionality over the varying numbers, positions and sizes of ground truth objects between images to supplementary data pre- and post-processing routines for handling occluded, truncated or distant (and thus "small") objects. While OD is, as stated, often a combination of regression and classification, we focus on its regression part, i.e., object localization. Predicting positional distributions instead of point estimates (Hall et al., 2020) is particularly beneficial as detected objects (and thus potential prediction errors) are typically propagated to subsequent algorithmic tasks, in the case of autonomous mobility systems, for instance, the prediction of trajectories for dynamic objects and motion planning for the ego-vehicle. Employing probabilistic models for these tasks that acknowledge the (potential) ambiguity of current object positions likely facilitates more informed predictions and thus safer operations.

The outlined challenges of OD have been addressed by specialized neural architectures like Faster R-CNN (Ren et al., 2015), YOLO (Redmon et al., 2016), SqueezeDet (Wu et al., 2017) and RetinaNet (Lin et al., 2017b) that are based on convolutional layers (see Section 2.2) and that are more complex compared to the MLPs used in the previous empirical evaluations. In particular, we consider the SqueezeDet model (Wu et al., 2017), a compact, fully convolutional neural network and, as a first step, adapt the Wasserstein-dropout objective to it (see the following paragraph). Next, we introduce the six considered OD datasets and sketch central technical aspects of training and inference. Since OD

[14] Apart from such standard applications of OD, various less obvious OD use cases exist that range from the OD-based counting of animal and tree populations on satellite images (Xue et al., 2017; Brandt et al., 2020) to the detection of items (Cai et al., 2021) that are taken from the shelves in supermarkets to enable, for example, checkout-free billing.

networks are often employed in open-world applications (like autonomous vehicles or drones), they likely encounter various types of concept shifts during operations. In such novel scenarios, well-calibrated self-assessment capabilities help to foster safe functioning. We therefore evaluate Wasserstein-SqueezeDet not only in-domain but on corrupted and augmented test data as well as on other object detection datasets. Lastly, we conduct comparable analyses for the larger RetinaNet architecture.

Architecture SqueezeDet takes an RGB input image and predicts three quantities: (i) $2D$ bounding boxes for detected objects (formalized as a $4D$ regression task), (ii) a confidence score for each predicted bounding box and (iii) the class of each detection. Its architecture is as follows: first, a sequence of convolutional layers extracts features from the input image. Next, dropout with a drop rate of $p = 0.5$ is applied to the final feature representations. Another convolutional layer, the ConvDet layer, finally estimates prediction candidates. In more detail, SqueezeDet predictions are based on so-called anchors, initial bounding boxes with prototypical shapes. The ConvDet layer computes for each such anchor a confidence score, class scores and offsets to the initial position and shape. The final prediction outputs are obtained by applying a non-maximum-suppression (NMS) procedure to the prediction candidates. The original loss of SqueezeDet is the sum of three terms. It reads $L_{\text{SqueezeDet}} = L_{\text{regres}} + L_{\text{conf}} + L_{\text{class}}$ with the bounding box regression loss L_{regres}, a confidence-score loss L_{conf} and the object-classification loss L_{class}. Our modification of the learning objective is restricted to the L2 regression loss,

$$L_{\text{regres}} = \frac{\lambda_{\text{bbox}}}{N_{\text{obj}}} \sum_{i=1}^{W} \sum_{j=1}^{H} \sum_{k=1}^{K} \sum_{\xi \in \{x,y,w,h\}} I_{ijk} \left[\left(\delta\xi_{ijk} - \delta\xi_{ijk}^{G} \right)^2 \right] , \tag{5.12}$$

with $\delta\xi_{ijk}$ and $\delta\xi_{ijk}^{G}$ being estimates and ground truth expressed in coordinates relative to the k-th anchor at grid point (i, j) where $\xi \in \{x, y, w, h\}$. See Wu et al. (2017) for descriptions of all other loss parameters. Applying W-dropout component-wise to this 4D regression problem yields

$$L_{\text{regres, W}} = \frac{\lambda_{\text{bbox}}}{N_{\text{obj}}} \sum_{i=1}^{W} \sum_{j=1}^{H} \sum_{k=1}^{K} \sum_{\xi \in \{x,y,w,h\}} I_{ijk} \left[\mathcal{W}(\xi_{ijk}) \right] , \tag{5.13}$$

where

$$\mathcal{W}(\xi_{ijk}) = \left(\mu_{\delta\xi_{ijk}} - \delta\xi_{ijk}^{G} \right)^2 + \left(\sqrt{\sigma^2_{\delta\xi_{ijk}}} - \sqrt{\left(\mu_{\delta\xi_{ijk}} - \delta\xi_{ijk}^{G} \right)^2 + \sigma^2_{\delta\xi_{ijk}}} \right)^2 \tag{5.14}$$

with $\mu_{\delta\xi_{ijk}} = 1/L \sum_{l=1}^{L} \delta\xi_{ijk}^{(l)}$ and $\sigma^2_{\delta\xi_{ijk}} = 1/L \sum_{l=1}^{L} \left(\delta\xi_{ijk}^{(l)} - \mu_{\delta\xi_{ijk}} \right)^2$ being the sample mean and the sample variance over L dropout predictions $\delta\xi_{ijk}^{(l)}$ for $\xi \in \{x, y, w, h\}$.

Tab. 5.4.: Basic statistics of the harmonized object detection datasets. Dataset size and number of annotated objects are reported for train data (second and third column) and test data (last two columns). For details on dataset harmonization, see the text in Section 5.4 and references therein.

	train		**test**	
dataset	# images	# objects	# images	# objects
KITTI	3,622	15,254	3,387	12,673
SynScapes	19,998	906,827	4,998	226,390
A2D2	22,731	121,320	4,186	36,544
Nightowls	30,064	50,225	6,595	10,766
NuImages	58,803	410,462	14,377	97,014
BDD100k	69,281	843,963	9,919	123,752

Datasets We train SqueezeDet networks on six traffic scene datasets: KITTI (Geiger et al., 2012), SynScapes (Wrenninge & Unger, 2018), A2D2 (Geyer et al., 2020), Nightowls (Neumann et al., 2018), NuImages (Caesar et al., 2020) and BDD100k (Yu et al., 2020). They differ from each other in dataset size (the large BDD100k dataset contains almost 20 times more images than the small KITTI dataset, see Tab. 5.4), time of day (Nightowls comprises only nighttime images) and data acquisition (SynScapes is simulation-based). For further information on the datasets, see Tab. B.5 (p. 162) in Appendix B.2.5. We employ image sizes of 672×384 and rescale all datasets (except for KITTI[15]) accordingly. To facilitate cross-dataset model evaluations (see paragraphs on OOD analyses in this section), we group the various object classes of the six datasets into three main categories: "pedestrian", "cyclist" and "vehicle" (see Tab. B.6 (p. 162) in Appendix B.2.5 for the object class mapping). Some static or rare object classes are discarded.

Technical aspects We compare MC-SqueezeDet, i.e., standard SqueezeDet with activated dropout at inference, with W-SqueezeDet that uses W-dropout instead of the original MSE regression loss. All models are trained for 300,000 mini-batches of size 20. After training, we keep dropout active and compute 50 forward passes for each test image. The detections from all forward passes are clustered using k-means (see Section 9.1 of Bishop (2006)).[16] The number of clusters is chosen for each image to match the average number of detections across the 50 forward passes. Each cluster is summarized by its mean detection and standard deviation. To ensure meaningful statistics, we discard clusters with 4 or less detections. The cluster means are matched with ground truth. We exclude predictions

[15] For KITTI, we crop images in x-direction to avoid strong distortions due to its high aspect ratio. In y-direction, only a minor upscaling is applied.

[16] Using the density-based clustering technique HDBSCAN (Campello et al., 2013) yields comparable results especially w.r.t. the relative ordering of the methods.

from the evaluation if their IoU with ground truth is ≤ 0.1. For each dataset, SqueezeDet's maximum number of detections is chosen proportionally to the average number of ground truth objects per image.

In-data evaluation To assess model performance, we report the mean intersection over union (mIoU) and RMSE (in pixel space) between predicted bounding boxes and matched ground truths. The quality of the uncertainty estimates is measured by (coordinate-wise) NLL, ECE, WS and ETL. Tab. 5.5 shows a summary of our results on train and test data for the KITTI dataset. The results for NLL, ECE, WS and ETL have been averaged across the 4 regression coordinates. MC-SqueezeDet (abbreviated as MC-SqzDet) and W-SqueezeDet (W-SqzDet) show comparable regression results in terms of RMSE and mIoU, with slight advantages for MC-SqueezeDet. At this point, we only consider versions of SqueezeDet that provide uncertainty scores. For a discussion regarding performance degradation w.r.t. the deterministic SqueezeDet (approximately 10%), see Appendix B.2.5. Considering uncertainty quality, we find substantial advantages for W-SqueezeDet across all evaluation measures. These advantages are due to the estimation of heteroscedastic aleatoric uncertainty during training (see also the test statistics "trajectories" during training for BDD100k in Fig. B.12 (p. 165) in Appendix B.2.5). The test RMSE and ECE values of all six OD datasets are visualized as diagonal elements in Fig. 5.9. The (mostly) "violet" RMSE diagonals for MC-SqueezeDet and W-SqueezeDet (top row of Fig. 5.9) again indicate comparable regression performances. Datasets are ordered by size from small (top) to large (bottom). The large NuImages test set occurs to be the most challenging one. Regarding ECE (bottom row of Fig. 5.9), W-SqueezeDet performs consistently stronger, see the "violet" W-SqueezeDet diagonal (smaller values) and the "red" MC-SqueezeDet diagonal (higher values). These findings qualitatively resemble those on the standard regression datasets and indicate that W-dropout works well on a modern application-scale network.

To analyze how well these OD uncertainty mechanisms function on test data that is structurally different from training data, we consider two types of out-of-data analyses in the following: first, we study SqueezeDet models that are trained on one OD dataset and evaluated on the test sets of the remaining five OD datasets. A rather semantic OOD study as features like object statistics and scene composition vary between training and OOD test sets. Second, we consider networks that are trained on one OD dataset and evaluated on corrupted versions (defocus blur, Gaussian noise) of the respective test set, thus facing changed low-level features, i.e., less sharp edges due to blur and textures overlayed with pixel noise, respectively.

Out-of-data evaluation on other OD datasets We train one SqueezeDet on each of the six OD datasets and evaluate each of these models on the test sets of the remaining five datasets. The resulting OOD regression scores and OOD ECE values are visualized as off-diagonal elements in Fig. 5.9 for MC-SqueezeDet (left column) and W-SqueezeDet (right column). Since datasets are ordered by size (a rough proxy to dataset complexity),

Tab. 5.5.: Regression performance and uncertainty quality of SqueezeDet-type networks on KITTI data. W-SqueezeDet (W-SqzDet) is compared with the default MC-SqueezeDet (MC-SqzDet). The values of NLL, ECE and WS are aggregated across their respective four dimensions, for details see Appendix B.2.5 and Tab. B.7 (p. 155) therein.

	train		**test**	
measure	MC-SqzDet	W-SqzDet	MC-SqzDet	W-SqzDet
mIoU (↑)	**0.705**	0.691	**0.695**	0.694
RMSE (↓)	**8.769**	9.832	14.666	**14.505**
NLL (↓)	8.497	**2.770**	25.704	**6.309**
ECE (↓)	0.615	**0.193**	0.825	**0.433**
WS (↓)	1.421	**0.315**	2.831	**0.900**
$\text{ETL}_{0.99}$	22.358	8.853	42.101	18.223

the upper triangular matrix corresponds to cases in which the evaluation dataset is especially challenging ("easy to hard"), while the lower triangular matrix subsumes easier test sets compared to the respective i.i.d. test set ("hard to easy"). Accordingly, we observe (on average) lower RMSE values in the lower triangular matrix for both SqueezeDet variants. The ECE values of W-SqueezeDet are once more smaller ("violet") compared to MC-SqueezeDet ("red"). The ECE diagonal of W-SqueezeDet is visually more pronounced compared to the one of MC-SqueezeDet since uncertainty calibration is effectively optimized during the training of W-SqueezeDet. The Nightowls dataset causes a cross-shaped pattern, indicating that neither transfers of Nightowls models to other datasets nor transfers from other models to Nightowls work well. This behavior can be understood as the feature distributions of Nightowls' nighttime images diverge from the (mostly) daytime images of the other datasets. The high uncertainty quality of W-SqueezeDet is underpinned by the evaluations of NLL and WS (see Fig. B.11 (p. 163) and text in Appendix B.2.5).

Out-of-data evaluation on corrupted datasets In contrast to the analysis above, we now focus on "non-semantic" data shifts due to technical distortions. For each test set, we generate a blurred and a noisy version.[17] Two examples of these transformations can be found in Fig. B.10 (p. 163) in Appendix B.2.5. In accordance with previous results, W-SqueezeDet provides smaller ECE values compared to MC-SqueezeDet on most blurred and noisy test sets (see Tab. 5.6). We observe a less substantial deterioration of uncertainty quality for blurring compared to adding pixel noise, possibly because the latter one more strongly affects short-range pixel correlations that the networks rely on.

[17] We employ the `imgaug` library (`https://github.com/aleju/imgaug`) and apply defocus blur (severity of "1") and additive Gaussian noise (i.i.d. per pixel, drawn from the distribution $\mathcal{N}(0, 20)$), respectively.

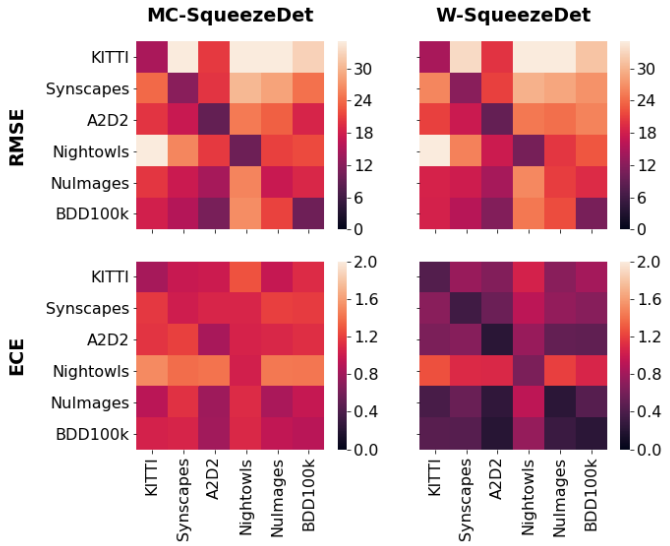


Fig. 5.9.: In-data and out-of-data evaluation of MC-SqueezeDet (l.h.s.) and W-SqueezeDet (r.h.s.) on six OD datasets. We consider regression quality (RMSE, top row) and uncertainty quality (ECE, bottom row). For each heatmap entry, the row label refers to the training dataset, the column label to the test dataset. Thus, diagonal matrix elements are in-data evaluations, non-diagonal elements are OOD analyses. W-SqueezeDet provides substantially smaller ECE values both in-data and out-of-data.

Complementary analysis of Wasserstein dropout for a larger OD architecture Apart from the compact SqueezeDet network, we conduct experiments on RetinaNet (Lin et al., 2017b), a more heavy-weight network that employs a so-called *focal* classification loss to compensate for imbalances between foreground classes and background. As a backbone network for feature extraction, RetinaNet uses a feature pyramid network (Lin et al., 2017a) that facilitates the detection of objects across multiple scales. RetinaNet is, like SqueezeDet, fully convolutional and based on anchor boxes, however, it employs a smooth-L1 loss instead of an L2 loss for box regression (see Section 2.3 for a discussion of the properties of these losses). As Wasserstein dropout is motivated as an extension of the standard L2 loss for label distributions, we employ an L2 loss instead of a smooth-L1 loss for the Wasserstein dropout-enhanced RetinaNet (L2-W-RetinaNet). Technically, it is constructed as described above for W-SqueezeDet. For better comparison with the smooth-L1-loss-based standard RetinaNet, we moreover consider a straight forward adaptation of Wasserstein dropout for the smooth-L1 loss, which yields a network termed Smooth-L1-W-RetinaNet. To extend the benchmarking beyond dropout-based uncertainty methods, several OD uncertainty estimation techniques from Feng et al. (2020) are considered, among others, loss attenuation, BayesOD, and pre-/post-non-maximum-suppression ensembles (see ibid. for details). While these techniques employ, at their core, standard mechanisms like parametric modeling and ensembling (and also dropout), they are refined for object

Tab. 5.6.: Out-of-data evaluation of MC-SqueezeDet (MC-SqzDet) and W-SqueezeDet (W-SqzDet) on distorted OD datasets. Each model is trained on the original dataset and evaluated on two modified versions of the respective test set: a blurred one (first two columns) and a noisy one (last two columns), see the text in Section 5.4 for details. We report the expected calibration error (ECE) and find W-SqueezeDet to perform better than MC-SqueezeDet on most datasets.

	defocus blur		**Gaussian noise**	
dataset	MC-SqzDet	W-SqzDet	MC-SqzDet	W-SqzDet
KITTI	**1.034**	1.082	**1.021**	1.084
SynScapes	1.081	**0.503**	0.941	**0.910**
A2D2	0.921	**0.295**	1.143	**0.617**
Nightowls	1.067	**0.803**	0.992	**0.682**
NuImages	0.908	**0.332**	**0.760**	0.849
BDD100k	1.012	**0.390**	0.833	**0.633**

detection and vary, among others, in the way the networks' final uncertainty estimates are obtained from the respective per-anchor uncertainty estimates. In Feng et al. (2020), the best uncertainty quality is obtained for a method termed *BayesOD+dropout* that combines MC dropout and BayesOD (Harakeh et al., 2020), a technique that in turn combines parametric uncertainty estimates per anchor (Truong-Le et al., 2018) with a specific object proposal merging strategy (see Feng et al. (2020) for details). To foster fair comparison with this method, we implement L2-W-RetinaNet and Smooth-L1-W-RetinaNet within the code base[18] accompanying Feng et al. (2020). Training all these probabilistic object detectors on the large BDD100k dataset (see above), we find that both Smooth-L1-W-RetinaNet and L2-W-RetinaNet yield good results for regression ECE (0.21 and 0.20, respectively), WS (0.18 and 0.17, respectively) and ETL (5.82 and 5.74, respectively). In these regards, they outperform the considered benchmark approaches. BayesOD+dropout is the strongest of these competitors and provides an ECE value of 0.33 while being (approximately) on par with the W-dropout-based uncertainty methods for WS and ETL. Evaluating these models on OOD data, namely, on the conceptually broad NuImages dataset (see above), the relative ordering of the uncertainty methods is preserved, expectedly, however, on a lower absolute quality level. The results are in accordance with the ones obtained for 1D standard regression datasets and SqueezeDet and emphasize the stability and uncertainty estimation quality of Wasserstein dropout that render it beneficial for a broad range of, even complex, deep learning architectures. While the RetinaNet experiments indicate that W-dropout can be combined with architecture-

[18] `https://github.com/asharakeh/pod_compare`.

specific regression losses (smooth-L1 loss), preliminary experiments (that are not part of this thesis) hint for a similar adaptability of our uncertainty method for different object proposal matching strategies.

5.5. Discussion

The prevailing approaches to uncertainty quantification rely on parametric uncertainty estimates by means of a dedicated network output. In this chapter, we propose a novel type of uncertainty mechanism, *Wasserstein dropout*, that quantifies (aleatoric) uncertainty in a purely non-parametric manner. By revisiting and newly assembling core concepts from existing dropout-based uncertainty methods, we construct distributions of randomly drawn sub-networks that closely approximate the actual data distributions. This is achieved by a natural extension of the Euclidean metric (for points) to the 2-Wasserstein metric (for distributions). In the limit of vanishing distributional width, i.e., vanishing uncertainty, both metrics coincide. Assuming Gaussianity and making a bootstrap approximation, the Wasserstein metric can be replaced by a compact loss objective affording stable training. To the best of our knowledge, W-dropout is the first *non-parametric* method to model heteroscedastic aleatoric uncertainty in neural networks. It outperforms the ubiquitous parametric approaches, as, e.g., shown by our comparison to deep ensembles (PU-DE). We moreover derive, inspired by residual learning, a Gaussian-likelihood one-sample (GL-OS) variant of Wasserstein dropout that encodes uncertainty and model error information, unlike standard Wasserstein dropout, not only in the spread of the sub-network ensemble but mostly in the distance between deterministic network and the mean output value of the stochastic sub-networks. The GL-OS variant yields competitive uncertainty estimates and requires less training compute, however, at the price of a bimodality artifact in the induced output distributions that is due to the one-sample approximations. In particular due to this artifact, the focus of the subsequent systematic analyses is put on the standard Wasserstein dropout technique.

Comparing the uncertainty estimates that standard Wasserstein dropout induces *under data shift* with those of deep ensembles (PU-DE) and parametric models combined with dropout (PU-MC), further advantages of our modeling technique are revealed: the Wasserstein-based technique still provides (on average) better calibrated uncertainty estimates while coming along with a higher stability across a variety of datasets and data shifts. In contrast, we find parametric uncertainty estimation (PU) to be prone to instabilities that are only partially cured by the regularizing effects of explicit or implicit (dropout-based) ensembling (PU-DE, PU-MC). With respect to worst-case scenarios, W-dropout networks are by a large margin better than either PU-DE or PU-MC. This makes W-dropout especially suitable for safety-critical applications like automated driving or medical diagnosis, where (even rarely occurring) inadequate uncertainty estimates might lead to injuries and damage. Furthermore, while our theoretical derivation focuses on aleatoric uncertainty, the presented distribution shift experiments suggest that W-dropout

is also able to capture epistemic uncertainty. Finding a theoretical explanation for this may be the subject of future research.

Concerning computational demands, W-dropout is roughly equivalent to MC dropout and, in fact, could be used as a drop-in replacement for the latter. While L-fold sampling of sub-networks increases the training complexity, we observe an increase in training time that is significantly below L in our implementation. Inference is performed in the same way for both methods, and thus, their run-time complexities are equivalent. Compared to deep ensembles, W-dropout's use of a single network reduces training and storage requirements at the expense of multiple forward passes during inference. This property is shared with MC dropout, and approaches exist to reduce the prediction cost; for instance, last-layer MC dropout allows sampling-free inference (see also Postels et al. (2019)).

In addition to the toy and 1D regression experiments, SqueezeDet is selected as a representative of object detection networks. We find the above-mentioned properties of Wasserstein dropout to carry over to Wasserstein-SqueezeDet, namely, the enhanced uncertainty quality and the increased stability under different types of data shift. At the same time, the observed performance losses are minimal. Overall, our experiments on SqueezeDet show that W-dropout scales to larger networks that are relevant for practical applications. Complementary evaluations using the RetinaNet architecture confirm these results.

When intending to employ uncertainty estimation as a safeguard against model errors, the distributional properties of the normalized residuals gain importance. To address such properties, we introduce the ETL as a measure for rare and critical cases where uncertainty is strongly underestimated. While we find that W-dropout leads to more Gaussian residuals compared to our benchmarks, we still observe remaining deviations. A priori, it is not clear whether the aleatoric uncertainty in complex data is Gaussian or whether such rare cases could be better described with heavier-tailed distributions. If this is the case, the question arises of whether dropout mechanisms are flexible enough to model distributions outside the Gaussian regime, which we investigate in Section 4.2.

Taking a step back, the idea of exchanging the distributions allows us to apply the framework to a variety of tasks beyond regression and makes the migration from single-point modeling to full distributions a rather general concept. Replacing Gaussians with Dirichlet distributions makes an application to classification conceivable, where Malinin & Gales (2018) employ parametric (Dirichlet) distributions to quantify uncertainty. Conceptually, our findings suggest that distribution modeling based on *sampling* generalizes better compared to parameterized counterparts. An observation that might find applications far outside the scope of uncertainty quantification.

6. Building Uncertainty Estimators for Product-Grade Deep Learning Systems

Various technical approaches exist to realize uncertainty estimation in NNs (see Section 3.2 for an overview). Depending on the application context, these uncertainty techniques serve different purposes. MC dropout, for example, which we studied in Section 4.2, is well suited to "recognize" whether statistical concepts in previously unseen inputs are (un-)familiar to the network in terms of epistemic uncertainty (see Section 3.1 for an overview of uncertainty sources). For our method, Wasserstein dropout (see Chapter 5), we extended MC dropout to bootstrapped data distributions. This way, it moreover captures other forms of uncertainty, namely, heteroscedastic aleatoric uncertainty (see Section 3.1).

In this chapter, we intend to abstract from the technical inner workings of uncertainty mechanisms (as studied previously) and, instead, take an application perspective. We consider uncertainty techniques as "tools", as pre-made building blocks, that come along with specific combinations of strengths and weaknesses. Existing mechanisms vary, for instance, not only in their ability to capture different types of uncertainty (as mentioned above) but also in their stability under data shifts—a critical property when used as part of real- and open-world learning systems. Apart from stability on structurally new data, these systems typically involve a variety of further requirements regarding robustness but also beyond. Making these requirements explicit and closely addressing them is of particular importance when deploying these systems at scale in safety-critical contexts (think, for instance, of autonomous driving). To this end, we propose a framework that, firstly, structures and shapes ML application demands, secondly, guides the selection of a suitable uncertainty estimation method, and, thirdly, provides strategies to validate this choice and to uncover structural weaknesses.

To illustrate application demands, we consider two central classes of use cases: *uncertainty for safety* (e.g., Truong-Le et al. (2018); Shafaei et al. (2018); Aravantinos & Schlicht (2020)) and *uncertainty for (semi-)automation* (e.g., Thrun et al. (2005); Giannetti (2017); Sünderhauf et al. (2018); Papananias et al. (2019)), respectively. The already-mentioned safety-critical applications (uncertainty for safety) comprise use cases that range from recommendation engines, e.g., in a medical context, to cyber-physical systems such as robots, drones, or vehicles. A domain expert's notion of a "good" uncertainty mechanism in these fields is likely given by a high-level requirement, such as "do not provide a semantic segmentation of a computed tomography (CT) image unless you are 99.5% certain about it" (in a medical application) or even more abstract, such as "contribute to defensive driving and compliance with the traffic law" (in the case of an automated vehicle).

Apart from these safety-critical applications, uncertainty quantification is crucial for (semi-)automation, e.g., when phasing in ML systems for quality assurance (QA) in an industrial production plant or to (partially) automate the classification of texts (Lewis & Gale, 1994). For QA purposes, an image-processing model could decide whether or not the surface of a (mass) product is sufficiently homogeneous and undamaged. If the model is uncertain regarding its decision, the product is inspected by a human expert to ensure high quality standards. A practitioners' requirement regarding this QA model might be that "uncertainty estimates enable us to flexibly maximize the degree of automated quality inspection such that less than 1 in 10,000 products leaves the plant despite being in a defect state". In the case of an automated text annotator one might similarly ask for "accurate uncertainty estimates that allow us to hand over difficult examples to a human annotator such that the results of the ML annotation tool (with occasional human overrule) have the same quality as the ones of a fully manual annotation process".

The above-outlined practitioner's view on uncertainty estimators varies from the perspective of ML researchers putting forward novel concepts to modeling and testing uncertainty estimates. Not only does a complexity gap occur between lab-oriented scientific proposals and actual industrial applications, the practitioner's focus is moreover on the alignment of the actual technical state-of-the-art with a desired state, i.e., the respective application requirements. Such a systematic alignment is a prerequisite for responsibly deploying ML systems, i.e., in a way that mitigates their risks and thus allows their strengths to play out safely. We contribute to this alignment by providing a *framework for the development and testing of neural uncertainty estimators based on high-level requirements*, aiming at ML applications with tailored uncertainty mechanisms. By doing this, we bridge the gap between, on the one hand, industry researchers and safety engineers who develop standards and set high-level requirements for safe ML such as the sub-committee 42 of ISO's and IEC's joint technical committee 1 (ISO/IEC JTC 1/SC 42, 2017) and, on the other hand, scientific ML researchers developing uncertainty mechanisms and testing algorithms. To the best of our knowledge, this is the first work that systematically links these two perspectives on neural uncertainty estimation.

Developing and testing learned neural models is qualitatively different from developing and testing (traditional) complex systems that are manually assembled. This becomes evident when comparing three complex objects, each composed of tens of thousands smaller units: a car, programmed software and a neural network. While we understand the role of a battery, an electric motor and tires in a car (and the classes and objects in an object-oriented software solution) and how they interplay, such knowledge about the semantic inner workings is inherently inaccessible for a neural network. It is a non-interpretable black box that is optimized using learning algorithms.

These properties of neural models necessitate novel approaches to testing. While traditional hardware and software systems have been specified (Dwyer et al., 1999; Tahat et al., 2001; Grunske, 2008; Pandey et al., 2010; Meth et al., 2015) and tested (Kropp et al., 1998; Tahat et al., 2001; Nebut et al., 2003; Majzoobi et al., 2008; Klein et al., 2009; Kavitha et al., 2010; Esteve et al., 2012) for decades, sometimes even with the help of

mathematical proofs (Hartman, 2006), respective approaches (often) rely on splitting a complex system into smaller semantic units (e.g., unit testing to debug software). Smaller units are easier to safeguard, their roles (in a logical or physics-based cause-and-effect model) and failures are easier to understand and can either be mitigated by understanding (and removing) the cause or by adding redundancy as a fail-safe. A deep neural network (DNN) on the contrary is not composed of semantic building blocks and thus does not become safer by adding an additional (or redundant) layer. Moreover, the (effective) input spaces in classical engineering (e.g., the forces and torques a machine part is exposed to) are small compared to the (often) million-dimensional input spaces, e.g., an image-processing ML model operates on. Such smaller input spaces allow for more comprehensive testing, i.e., sampling the input space in ways such that most application scenarios can be seen as interpolations of these sample points. Furthermore, the physics-based (or logical) models that describe such assembled systems ensure a certain stability and thus predictability of the system behavior in these interpolation cases. This is in contrast to (many) ML systems that are optimized on extremely sparse data samples of high-dimensional spaces such that distinctions between inter- and extrapolation are hardly possible. These purely data-based models provide moreover less stable generalization as their outputs are highly susceptible to certain types of small changes in input space (see adversarial attacks (Goodfellow et al., 2015)).

This combination of non-interpretable models and high-dimensional input spaces renders it hardly possible to exhaustively test semantic and high-level requirements for real-world ML models. Nevertheless, *testing does yield insights into multiple aspects of the qualitative and quantitative behavior of an ML function* within a given computing and cost budget.

While these considerations hold for neural models in general, they also apply in the special case of developing and testing neural uncertainty estimators that are integrated into deep learning (DL) models. Accurate uncertainty estimators provide an implicit description of the model's domain of proper functioning, which makes them promising tools for safe ML. Testing uncertainties therefore differs from ML performance testing (e.g., of a specifically designed layout of a hidden layer) not only by the considered metrics (e.g., uncertainty calibration measures that are "orthogonal" to performance metrics) but also in the data selection strategies (by putting a focus on transitions to out-of-data or safety-critical scenarios).

In this chapter, we provide a framework for systematically breaking down high-level requirements onto uncertainty modeling techniques and uncertainty test cases. Following the top-down structure of the framework ensures compatibility with many requirement-based development approaches used for non-ML systems and components. In detail, we contribute

- by categorizing potential requirements, for instance w.r.t. uncertainty quality, generalization ability (e.g., extrapolation) or technical aspects (e.g., when building upon previous systems),

- by analyzing how and to which extend requirements can be addressed by the differing paradigms of uncertainty modeling and thus, which type of uncertainty estimator to choose for a given DL use case,
- by providing a hierarchy of model tests to evaluate whether the uncertainty requirements are met by the selected estimator, and whether systematic weaknesses exist that might hinder its further use or deployment.

This chapter is organized as follows: first, we contextualize our work in the fields of trustworthy ML and ML testing in Section 6.1. Next, we lay out the structure of our uncertainty framework in Section 6.2. Caveats regarding neural uncertainty estimators and best practices on how to address them are presented for each of the five steps of the framework in Sections 6.3 to 6.7. An outlook in Section 6.8 concludes the chapter.

6.1. Regulatory and technical perspectives on trustworthy ML

As a technical tool to establish safe ML systems, uncertainty estimation may form a cornerstone to satisfy demands of upcoming ML regulations and standards some of which are outlined in the first paragraph. Next, we sketch already existing approaches to operationalize trustworthy ML as well as challenges that practitioners face when building reliable ML systems (second paragraph). Subsequently, algorithmic techniques that increase the reliability of learned systems are reviewed (third paragraph).

Trustworthy ML from a regulatory perspective Both governmental institutions and industry are about to set rules for ML systems. Aiming for an effective protection of basic rights in an era of ubiquitous ML systems, several countries put forward law initiatives and guidelines for regulating ML models. In November 2020, the White House released guidelines for public authorities in the US on how to regulate artificial intelligence (AI) systems (Executive Office of the U.S. President, 2020). In April 2021, the European commission published a "proposal for a regulation laying down harmonized rules on AI" (European Commission, 2021) that (among others) classifies and handles ML applications according to their "criticality". The proposal moreover asks for technical documentations that detail how such critical ML systems function. Aside these official regulations (and, if applicable, in accordance with them), industry and standardization bodies define minimal requirements, interfaces and liability regimes as prerequisites for functioning technological markets. International standards are, e.g., developed by the sub-committee 42 of ISO's and IEC's joint technical committee 1 (ISO/IEC JTC 1/SC 42, 2017) that issued technical reports, e.g., on the trustworthiness in AI (ISO/IEC TR 24028:2020, ISO/IEC JTC 1/SC 42 (2020)) and on the assessment of the robustness of neural networks (ISO/IEC TR 24029-1:2021, ISO/IEC JTC 1/SC 42 (2021a)). Actual standards, e.g., for AI management systems (ISO/IEC CD 42001, ISO/IEC JTC 1/SC 42 (2021b)) are under way. These international initiatives build on standardization efforts at the national level, see, e.g., the German standardization roadmap on AI (Wahlster &

Winterhalter, 2020), and harmonize them. Moreover, application-specific norms come into existence, for instance, the ANSI/UL 4600 standard for safety for the evaluation of autonomous products (Underwriters Laboratories, 2020) that, among others, targets applications in the field of autonomous mobility. For the specific context of safe AI for road vehicles, see ISO TC 22/SC 32 (2021).

Trustworthy ML from a practitioners' perspective When developing industry-scale ML applications from (newly invented) ML modeling techniques, corporate researchers and practitioners face challenges that are often not addressed by basic research in machine learning: Holstein et al. (2019), for instance, survey ML product teams w.r.t. fairness in ML and identify challenges such as the unavailability of both high-quality datasets and tools for fairness-focused debugging. Broadening the focus, Lwakatare et al. (2020) provide a literature review of obstacles such as adaptability and scalability that ML practitioners encounter when developing and maintaining ML-based systems. Similarly, Ishikawa & Yoshioka (2019) interview ML engineers to identify ML-specific challenges from a software-engineering perspective. Sculley et al. (2015) compile ML-specific risk factors, e.g., boundary erosion and hidden feedback loops, that are likely to cause technical debt when operating application-scale ML models.

At the same time, applied researchers and practitioners propose hands-on approaches to operationalize trustworthy ML. For instance, the concept of model reporting by means of model cards (Mitchell et al., 2019), and comparably, the idea of providing fact sheets that outline relevant attributes of an ML service (Arnold et al., 2019). Operationalizations of trustworthy ML for safety-critical systems take these approaches a step further and require specific (statistical) evidences (e.g., quantitative tests, see Abrecht et al. (2021)) that are bound together by an overarching safety argumentation which motivates their setup and configuration (Koopman et al., 2019; Mock et al., 2021). An important example for such structured, qualitative argumentations are so-called safety assurance cases (that have their roots in classical safety engineering, see ISO standard 15026 (ISO/IEC JTC 1/SC 7, 2019)). For concrete approaches to mitigate safety concerns (and thus contributing to safety argumentations), see, e.g., Koopman & Osyk (2019) and Willers et al. (2020).

Algorithmic approaches to trustworthy ML On a technical level, the trustworthiness and reliability of models is fostered by a broad range of machine learning techniques. Apart from uncertainty estimators, this entails: interpretability methods (on the level of pixels (Selvaraju et al., 2017; Brendel & Bethge, 2019) or semantic concepts (Kim et al., 2018)), mechanisms to enhance (individual or group) fairness (e.g., Adel et al. (2019)) and techniques to reduce the susceptibility of ML models w.r.t. adversarial attacks (e.g., by means of robust training (Madry et al., 2018)). For a survey on practical methods for ML safety, see, e.g., Houben et al. (2022).

Systematically evaluating the effectiveness of such safe-ML components is crucial to determine whether they are sufficient w.r.t. high-level requirements. Extensive surveys of the broad body of work on ML testing are provided by Zhang et al. (2020a), Riccio

et al. (2020), Braiek & Khomh (2020), and Corso et al. (2021). Of those, Zhang et al. (2020a) analyze testing properties, testing components and testing workflows and identify (among others) a lack of research on how different assessment metrics correlate with one another and with the test's ability to uncover faults of the model, respectively. Numerous algorithmic tools to unveil (and debug) model weaknesses were put forward, e.g., DeepXplore (Pei et al., 2017) and TensorFuzz (Odena et al., 2019) both of which adapt coverage-guided techniques from software testing to neural models.

Making implementations of such algorithmic approaches publicly available, e.g., in the form of open-source toolkits, contributes to their widespread application. Noteworthy examples for uncertainty toolkits comprise *UQ360* (Ghosh et al., 2021) and *Uncertainty-Toolbox* (Chung et al., 2021a) that ease the usage and evaluation of (custom) uncertainty estimators and foster their fair comparison. While being promising, for now, these toolkits do not contain methods that are specifically developed for more complex architectures like convolutional and recurrent DNNs and support task-specific approaches and measures (e.g., for autonomous driving (AD)) only in a limited fashion.

6.2. A framework for developing and testing neural uncertainty estimators

While established frameworks for requirement-based development and testing (see e.g., ISO/IEC JTC 1/SC 7 (2014); ISO/TC 22/SC 32 (2018)) provide a hull that remains applicable to ML systems, these learned models require novel technical realizations (see introduction). For the task of estimating uncertainties in neural networks, we analyze these conceptual challenges and propose best practices for tackling them. While the choice of the DNN architecture is, of course, central to the successful utilization of ML, we focus on the self-assessment capabilities of the learning system, as they are, to a certain degree, more agnostic to the task at hand. Based on our practical observations, the features and development processes of ML models are typically driven from two sides: on the one hand, from an application-oriented direction governed by the needs and requirements of a given (or future) product. On the other hand, by dedicated ML researchers and developers who are focused on feasibility and the state-of-the-art in the field. The proposed development and testing framework is intended as a structural aid for the interplay between these points of view.

For our purposes, subdividing the framework into five steps provides sufficient granularity. These steps and their intermediate results are visualized as a flow chart in Fig. 6.1. In the following, we describe the purpose of each step and name some of their key elements:

1. First, we grasp the *initial demand for an uncertainty estimator* by means of high-level (e.g., textual) requirements. Understanding the intended purpose as well as the technical modalities of the underlying ML system poses the basis for uncertainty modeling and testing. Moreover, the operational domain of the uncertainty estimator is specified, i.e., the set of inputs on which it is supposed to function properly.

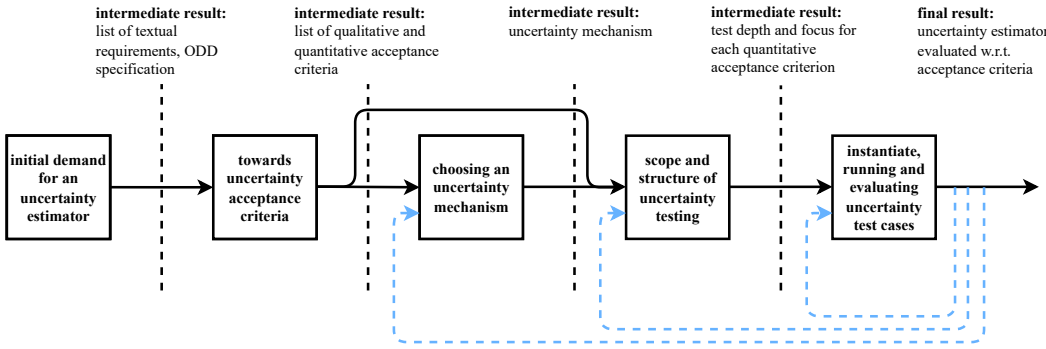


Fig. 6.1.: A structured approach to obtain an uncertainty estimator that is tailored to a given deep learning application. The proposed framework is subdivided into five steps in which uncertainty requirements are collected and quantified (steps 1 and 2), an uncertainty mechanism is selected or constructed (step 3) and the quality of its uncertainty estimates is systematically tested (steps 4 and 5), see the following sections for details. The uncertainty acceptance criteria (step 2) provide the basis for both uncertainty modeling (step 3) and test strategy (step 4). Inconsistent or failing tests may entail changes of the test cases, of the testing strategy or of the uncertainty mechanism (see light blue dashed backward arrows).

2. Next, we structure and refine the high-level requirements by means of requirement categories that we lay out. Heading toward *measurable (and thus testable) acceptance criteria*, the quantitative requirements are "translated" into 3-tuples of semantic data specification, measure of uncertainty quality and threshold value.

3. We then *choose an uncertainty mechanism* that seems suitable to meet the acceptance criteria, based on mappings of mechanism-specific characteristics onto the requirement categories. Apart from the complexity of the mechanism, its ability to model the predominant types of uncertainty is a key property.

4. Given both uncertainty acceptance criteria and uncertainty mechanism, we *set the scope and structure of uncertainty testing*. To this end, we introduce a hierarchy of tests that builds on semantic and non-semantic data selection strategies. For each acceptance criterion, a testing depth in this hierarchy is determined and focus points of testing are chosen.

5. Based on the selected testing focuses for each acceptance criterion, we finally *instantiate, run and evaluate uncertainty test cases*. This requires to select concrete test datasets and, e.g., specific search strategies and their initial parameters. Once executed, the binary results of the test cases are aggregated to decide whether the uncertainty estimator fulfills the uncertainty acceptance criteria.

In the following sections, we provide detailed information on each of the five steps of the framework.

6.3. Initial demand for an uncertainty estimator

Many factors, for instance regulatory contexts and technical specifications, influence uncertainty modeling. In Subsection 6.3.1, we study how these boundary conditions affect (directly or indirectly) the desired properties of an uncertainty estimator. The underlying ML use case moreover determines the operational design domain (ODD), i.e., the sets of inputs on which the uncertainty estimator is supposed to function (see Subsection 6.3.2). These input sets are described by semantic dimensions that need to be identified. Technical specifications may pose additional constraints, e.g., on the uncertainty estimator's depth of integration into the ML model (see Subsection 6.3.3). These constraints and the desired uncertainty properties from Subsection 6.3.1 pose requirements that guide the choice of the uncertainty estimation technique in a later step of the framework.

6.3.1. Purpose and desired properties of an uncertainty estimator

We broadly distinguish between external and internal purposes of an uncertainty estimator. External purposes range from regulatory requirements over industry standards to company guidelines. Alternatively, they may originate from a surrounding ML system or the need for backward compatibility. Such external purposes are likely to pose additional constraints on modeling and validating uncertainty functionalities. One may think of regulations of the financial industry where detailed technical guidelines for quantitative modeling exist, limiting, e.g., the use of complex model classes.

Internal purposes, in contrast, may be the goal to reach better functional performance (uncertainty for performance) or to employ well-calibrated uncertainty "sensors" that could be used to propagate uncertainty information along a chain of ML models or to initiate a hand-over to a backup system or a human (remote) operator. Such internal purposes likely go along with less restrictions on modeling compared to external purposes.

Uncertainty estimators are moreover employed as *passive* or *active* system components. Passive components monitor the state of the ML system, collecting information that is, e.g., used for long-term system optimization without having a direct impact on the recent situation. Active uncertainty components, on the contrary, may trigger subsequent actions or changes of the system behavior. The desired properties of these types of uncertainty estimators may vary accordingly: for the passive observer, a high sensitivity to abrupt changes or extreme events might be less relevant as long as the uncertainty estimates are *on average* accurate and bias-free. A dynamic bias, however, might be welcome for an active uncertainty component as, e.g., a slight overestimation of the uncertainty could provide an additional safety margin. While such an uncertainty actor might be inactive for most input scenarios, it is expected to act accurately in *individual* critical scenarios (as opposed to an on-average accurate behavior, see above). In case uncertainty estimates are used to safeguard against several types of undesired model behavior (e.g., false positives

and false negatives in a classification setting), the individual "risk profile" of the use case determines how to balance them appropriately.

6.3.2. Operational design domain of the uncertainty estimator

Next, we approach the intended *operational design domain* (ODD) of the uncertainty mechanism, i.e., the areas of input space where it is supposed to work reliably (see, e.g., NHTSA (2017); Koopman & Fratrik (2019)). Specifying these areas is especially challenging in open-world settings where input data is not fully under the control of the operator. In the case of AD systems, for instance, examples for under-specified ODDs could be geographical ones like fixed routes (e.g., of a regular bus), lanes exclusively for automated traffic or geo-fenced areas as well as "situational" ones like driving under stop-and-go traffic conditions. However, specifying an ODD in a complex high-dimensional space is not only approximate and incomplete by its very nature, the specifications may moreover have varying degrees of detail and are sometimes even contradictory.

To at least partially account for this fundamental problem, we propose to determine the ODD of an uncertainty estimator by using two techniques that complement each other and that furthermore help to identify (potentially occurring) contradictions and inconsistencies: on the one hand, by setting semantic boundaries and, on the other hand, by a scenario-based (or edge-case-based) approach where various in-domain-, out-of-domain-, and "borderline" input scenarios are compiled. The former approach seeks to describe and threshold relevant semantic dimensions of the input space. Taking, for instance, an ML application that processes traffic images, such descriptions comprise scene parameters ranging from the type of traffic over lighting conditions to numbers and positions of traffic participants. One may furthermore build on existing ontologies or knowledge graphs in the considered field of application, if such representations are available and appropriate. The latter technique builds on collections of in-domain and out-of-domain input scenarios as an exploratory tool to sharpen and improve on the ODD as constructed above. Comparing these scenario sets with the identified semantic boundaries, may reveal wrongly or unspecified semantic dimensions, e.g., by analyzing whether the selected "borderline" scenarios are consistent with the chosen semantic thresholds.

One can, moreover, build on the ODD specification of the underlying DNN, if given. Typically, however, the uncertainty estimator is supposed to also function reliably outside the original (performance) ODD of the DNN (see Fig. 6.2). Put differently, the uncertainty ODD is often larger than the performance ODD that may still provide a reasonable ODD "core". Both the performance ODD and the areas of the uncertainty ODD not overlapping with it will be addressed in testing. For those non-overlapping areas of the uncertainty ODD, a focus of testing are estimates of epistemic uncertainty.

6.3.3. Modeling uncertainties

Technical specifications not only impact, as outlined, the desired properties and the operational design domain of an uncertainty estimator. Here, we investigate how they may

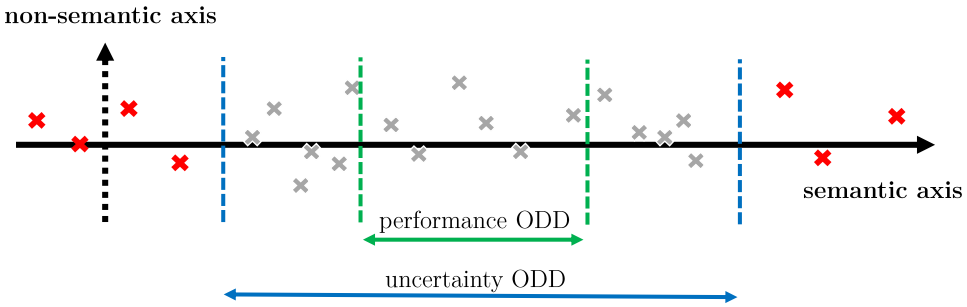


Fig. 6.2.: Schematic illustration of the *ODD of an ML model* (green interval) and the *ODD of its uncertainty estimator* (blue interval) in an input space that is sketched as a 2D plane. While the x-axis is considered to be semantically interpretable, the y-axis is non-semantic and thus not specifiable. The chosen semantic boundaries of the uncertainty ODD are wider compared to the model ODD, i.e., in this example it is supposed to function properly in a larger area of input space. Moreover, the complementary technique of compiling in-domain (gray crosses) and out-of-domain (red crosses) input points is in accordance with the chosen semantic borders in this toy illustration: no red crosses are within the uncertainty ODD and no gray crosses outside of it. The non-semantic y-coordinates of these input points cannot be specified and thus fluctuate randomly.

constrain uncertainty modeling, e.g., when using third-party ML components. Aside, we analyze how ML model chains determine the "flows" of uncertainty information through one of their models (see second paragraph). Lastly, output granularities of DL systems and uncertainty estimators are studied (third paragraph), as well as trade-offs and technical modeling constraints (last paragraph).

Depth of integration into the ML model To intertwine an uncertainty estimator with an ML model, the access restrictions the ML model is subject to need to be known as they impact both uncertainty modeling and testing. Such restrictions typically originate either from the state of model development (see next paragraph) or from black-box components with (to us) unknown architectures and weights that are employed in the ML system (see paragraph after next).

State of model development As the development of a model progresses from initial planning over engineering its architecture toward optimization and evaluation, a deep integration of an uncertainty mechanism into it becomes less likely: for a model under development, uncertainty modeling considerations may affect central building blocks of a DNN like architecture or objective function. An almost finalized ML model, on the contrary, that cannot be modified any longer, constrains uncertainty estimators to be pre- or post-processing units. Such a limitation may also result from requirements on backward compatibility that ask, e.g., to re-use previously employed model components.

Black-box components Many (premade) ML models underlie access restrictions, e.g., black-box models from third parties with (to us) unknown structure and parameter values. For such models a negative correlation between the extent of the restrictions and the depth of integration of the uncertainty estimator can be expected. As for finalized ML models (see above), one may rely on pre- or post-processing mechanisms or, for a model in a model chain (see below), on bypassing the uncertainty estimates from preceding ML components. In contrast to the state of model development, the use of black-box components with access constraints directly impacts testing as they exclude testing methodologies that require knowledge about the internal state of a DNN.

Flow of uncertainty information In many real-world ML systems, an ML model is not stand-alone but part of a model pipeline (see, e.g., Sculley et al. (2015)). Think, e.g., of a (prototypical) AD module stack that detects and characterizes vulnerable road users (VRUs), positions them in a 3D model of the recent traffic scene and predicts their future positions and velocities. Given the information flow along this model chain, various ways exist to enrich it with uncertainty information. In autonomous driving, for instance, this information flow may be pedestrian proposals that are passed to a pedestrian classification network or, further down the AD stack, the 3D phase space coordinates of a detected pedestrian that serve as input for a motion prediction model. Adding uncertainty information, the VRU detections may be accompanied by confidence estimates and predicted future VRU positions (the mean estimates) may go along with estimated standard deviations. For such *flows of uncertainty information* along a model chain, we provide a taxonomy (see Fig. 6.3) and discuss how the desired type of uncertainty flow impacts the development and testing of the uncertainty estimator.

We broadly distinguish between input, output and internally used uncertainties. Adding an uncertainty estimate to a model output (see a) and b) in Fig. 6.3) is a standard scenario of uncertainty quantification. The two sketches in Fig. 6.3 differ by the estimator's depth of integration into the considered (turquoise) model (see discussion in the previous paragraphs). While the estimator in a) is an integral part of the model, the estimator in b) is a post-processor, i.e., a calculation routine that extracts an uncertainty estimate from the model's standard outputs. It may, for instance, calculate the entropy of a classifier's softmax output that poses a simple measure of inter-class uncertainty. Alternatively, an already existing uncertainty flow could be re-calibrated, using routines like temperature or Platt scaling (Guo et al., 2017).

Incorporating input uncertainties into a model's uncertainty estimation is crucial for many practical applications as such a propagation of uncertainty allows us to eventually evaluate the behavior of an ML application as a whole, assuming a sufficiently high quality of the uncertainty estimates. Research on the propagation of uncertainties (Wright et al., 2000; Ji et al., 2019) has to deal with the limited availability of real-world datasets containing uncertainty-enhanced input features and the fact that such chains of ML models are highly application-specific, rendering them no typical object of study for a broad scientific community. For information-restricted black-box models, uncertainty

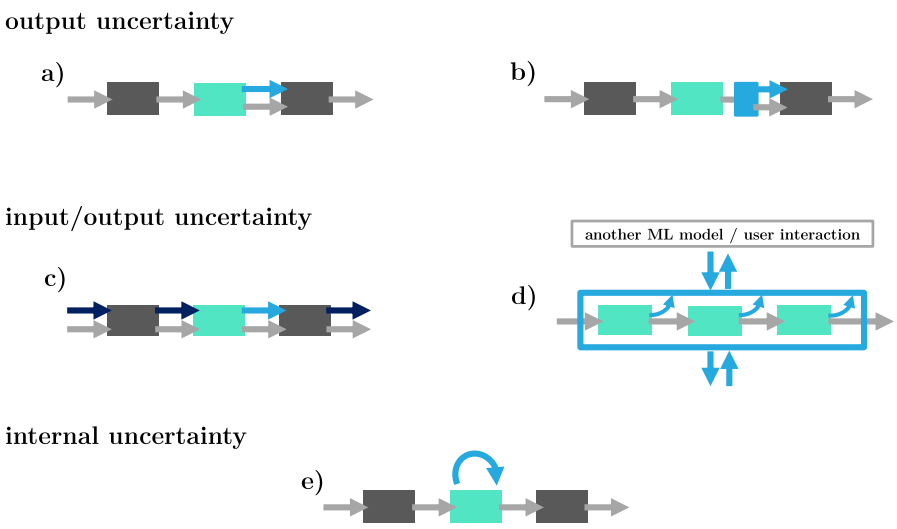


Fig. 6.3.: Prototypical "flows" of uncertainty information (blue arrows) along a chain of ML models (gray blocks and gray arrows). While uncertainty estimates for the entire chain are relevant (see d), we focus on uncertainty estimation for a single model in the chain (turquoise block, see a, b, c, e). In a) and b), output uncertainties are modeled and propagated to the subsequent model. The uncertainty estimator is integrated into the model (a) or a separate post-processing unit (b), respectively. An uncertainty estimator may incorporate propagated uncertainty information from preceding models (see c and d). Internal uncertainty estimates, on the contrary, are not propagated but generated and used within a model (see e), e.g., to improve the quality of its output.

propagation can be realized by bypassing uncertainty information, i.e., by circumventing the considered ML model and instead piping the uncertainty information directly from the previous model to the subsequent one in the chain. A conservative uncertainty margin for the black-box model might be added. Instead of modeling only one "segment" of this uncertainty flow (Fig. 6.3, c), some use cases require to zoom out and consider the entire model chain (Fig. 6.3, d). A critical capability of such a chain could be to incorporate external uncertainty information from surrounding software systems, e.g., to flexibly react to early indicators of a changing environment (such as a confusing traffic situation ahead). In the remainder of this chapter, however, we focus on one ML model in such a model chain.

Finally, some ML models employ uncertainty estimates internally, i.e., neither the model inputs nor its outputs carry uncertainty information and thus there is no uncertainty information flow along the chain. The uncertainty estimates are instead generated and processed within the ML model. In such cases, uncertainty information is typically used to improve the model's performance, either directly or indirectly, e.g., as a regularizer. Henaff et al. (2019) perform uncertainty-based regularization for trajectory prediction.

He et al. (2019) construct a two-stage object detection network whose region-proposal network uses a so-called var-voting scheme, i.e., final detections are obtained by location-confidence-based aggregation of neighboring detections. Preliminary detections with low uncertainty thus have a higher impact on the final detections.

Widening the focus, the flow of uncertainty information moreover influences quantitative testing (see Sections 6.6 and 6.7). Thoroughly evaluating, for instance, a setup like Fig. 6.3 c) requires the availability of realistic input uncertainty data, while a setup like Fig. 6.3 e) may have a focus on performance testing, rendering aspects like uncertainty calibration less important compared to most other uncertainty use cases.

Uncertainty granularity Given the types of uncertainty propagation presented above, we provide details on the kind of information that flows and its dynamics. Specifically, we compare the *operation modes* and the *output granularities* of the DNN and its uncertainty mechanism as they may diverge. For an image-processing DNN, for instance, that operates frame-wise and at a constant frame rate, the uncertainty estimator may generate estimates at another frame rate, change-based or solely for inputs that are structurally different from its training data. This holds true for output granularity where the desired granularity of uncertainty information could be higher or lower compared to the one of the DNN outputs. While, for instance, a segmentation network generates outputs on pixel level, uncertainty information might be required on the level of objects, image parts or entire images. Contrarily, one may think of a pedestrian detection network to be equipped with uncertainty estimates on body-part level. Such real-world ML systems often process structured information, i.e., both model inputs and outputs contain intrinsic (e.g., spatio-temporal) dependencies, think, e.g., of image-processing DNNs that predict scene graphs or semantic segmentations. Uncertainty modeling ought to reflect these dependencies, i.e., ideally generates uncertainty estimates that, e.g., change steadily along a semantic segment, as opposed to estimates that strongly fluctuate from pixel to pixel.

Trade-offs and technical constraints Requirements are often at odds with one another, e.g., the functional and the uncertainty-related requirements for an ML model. In most cases, uncertainty requirements must not be met at the expense of considerably decreased functional quality and at most small deteriorations are tolerable. Moreover, we might face trade-offs between uncertainty estimators and other safe-ML techniques (that foster, e.g., interpretability or fairness) as all of them share the same neural capacities. Further technical constraints for modeling uncertainties range from system complexity over storage to latency and vary strongly from use case to use case. Boundary conditions regarding time and resources not only apply to the uncertainty estimator itself but moreover to its development process where a simple “off-the-shelf” mechanism on the one hand and a scientifically novel custom estimator on the other hand mark extreme positions.

Results and remarks

The technical boundaries of the last section together with the above discussed aspects of intended functionality pose the high-level requirements the uncertainty estimator is subject to. All of them can be collected, either in an unstructured way or more formalized, using a requirement specification language (see, e.g., Reinkemeier et al. (2011)). While these requirements reflect the initial demand for uncertainty estimation (see Fig. 6.1), they reside on various levels of abstraction and thus require further concretization to base the development and testing of uncertainty mechanisms on them. In contrast, the operational design domain of the uncertainty estimator is already fully specified, i.e., relevant semantic dimensions have been determined and value ranges along these dimensions were set.

6.4. Toward uncertainty acceptance criteria

In this section, we take steps to derive formalized acceptance criteria from the above collected high-level requirements. For application-size ML systems, these requirements are often complex and thus "translate" into a variety of acceptance criteria. These "translations", like any modeling, go along with simplifications and assumptions that inevitably stress and neglect certain aspects of the original requirements. To reduce resulting blind spots, modeling ideally involves experts from different technical and non-technical domains (e.g., safety engineers, ML researchers and product owners). The diversity of the acceptance criteria is further increased by formulating them redundantly, i.e., by different experts who work independently of each other (also compare the qualitative "Swiss cheese" model in risk management (Perneger, 2005)). While these practices may increase the completeness of the acceptance criteria, it is beneficial to explicitly analyze whether completely fulfilled acceptance criteria actually capture the gist of the underlying requirements or whether they can be "hacked" (compare reward "hacking" in reinforcement learning (Hadfield-Menell et al., 2017)).

We help to structure and sharpen uncertainty requirements by providing a categorization of desirable uncertainty properties in Subsection 6.4.1. While selected qualitative requirements can already be considered as acceptance criteria, the quantitative ones should be further formalized. For those, we introduce a formal notation of acceptance criteria as 3-tuples consisting of (*semantic data specification*, *measure of quality*, *threshold value*) in Subsection 6.4.2.

6.4.1. Categorizing uncertainty requirements

Despite being case-specific at the level of details, qualitative and quantitative requirements for uncertainty estimation can be broadly categorized. We identify 10 categories, the first three of which capture different aspects of uncertainty quality. Categories four and five target conceptual properties of uncertainty estimation while categories six to nine focus on technical characteristics. They are complemented by a residual category for

application-specific requirements that do not fit elsewhere. In the following, we detail these requirement categories and highlight the benefits of fulfilling them:

Aspects of uncertainty quality

- **calibration** Uncertainty estimates are calibrated when their distribution matches the distribution of actual model errors (DeGroot & Fienberg, 1983; Naeini et al., 2015). These matchings are typically calculated across an entire dataset, rendering calibration a dataset-wide (or global) property. Global calibration is quantified by measures like ECE and desirable for both in-data (ID) inputs, i.e., inputs from within the ODD (or more concretely, from a dataset approximating it) and out-of-data (OOD)[1] inputs. A small ECE, e.g., assures that uncertainty estimates statistically resemble the model errors. While it does not imply that each individual large uncertainty estimate goes along with a large model error and vice versa, it allows us to assess the on-average quality of the uncertainty mechanism and provides a foundation for more detailed statistical analyses (see subsequent categories).

- **local calibration / heteroscedasticity** Uncertainty estimates are *locally* calibrated when they probabilistically match prediction confidence for each data point (see, e.g., Zhao et al. (2020)). This allows local inference on the model's trustworthiness for a given input. As most datasets do not provide ground truth uncertainties, local calibration is typically subject to resolution limits.[2] An assessable proxy measure is ECE calculated on (local) data subsets, e.g., on a subset of safety-critical scenes. A high ECE on such inputs could reveal weaknesses of an uncertainty estimator that remained undiscovered when calculating the ECE averaged over a large dataset as, e.g., underestimated uncertainties for safety-critical inputs might have been balanced by overestimated uncertainties for not safety-critical inputs. While even input-independent, constant uncertainty estimates could lead to small global ECE values (DeGroot & Fienberg, 1983), local calibration is closely linked to input-dependent, i.e., heteroscedastic, uncertainty estimates. Local calibration is desirable for ID data points and a variety of OOD scenarios. It is a strict criterion that implies global calibration and the ability to solve downstream tasks (see next category).

- **ability to solve downstream tasks** In the case that an uncertainty estimator is used as an active component in an ML system, its estimates are often employed as input for downstream decision making, especially whether safe and reliable operations of the system can be ensured on the current input data. The choice of the concrete downstream task to solve is highly application-specific and may, for instance, be the detection of inputs that are structurally different from training data (ID-OOD

[1] Be aware that the abbreviations ODD and OOD bear the risk of confusion. While ODD is the operational design domain, OOD stands for out-of-data, i.e., they refer to diverging concepts that are (in a qualitative sense) even contrary to one another.

[2] Assuming, however, a dataset that comes along with ground truth uncertainties $\sigma_{\mathrm{gt,i}}$, i.e., data points with labels $(\mu_{\mathrm{gt,i}}, \sigma_{\mathrm{gt,i}})$, and a model with predictions $(\mu_{\mathrm{i}}, \sigma_{\mathrm{i}})$, the model's local calibration could be evaluated, e.g., by means of a Wasserstein distance, i.e., as $\sum_i \left((\mu_{\mathrm{i}} - \mu_{\mathrm{gt,i}})^2 + (\sigma_{\mathrm{i}} - \sigma_{\mathrm{gt,i}})^2\right)$.

detector) to enable, e.g., for AD systems, safe stopping if necessary. Another example from automated perception is the recognition of false detections (e.g., of VRU reflections in windows) which are then no longer considered, e.g., for motion planning. In an industrial context, automated inspections that go along with high uncertainty estimates might trigger manual re-evaluations of individual workpieces. Technically, sufficiently strong correlations between uncertainty estimates and actual model errors are required to enable this kind of self-assessment. Compared to calibration (see above), less attention is paid to specific distributional properties of the uncertainty estimates. Instead, they are employed as a tool that often remains useful even when some desirable distributional properties are not met.

Conceptual properties

- **argumentatively substantiated** Broad scientific analyses, both theoretical and empirical, help to draw a realistic picture of the strengths and weaknesses of an uncertainty mechanism. On the one hand, theoretical justification allows for a more thorough understanding of the mechanism and thus enables more confident predictions of its behavior (e.g., on previously unseen data). In the long run, these investigations aim for theoretical verifications of DNN-based estimates (see, e.g., Salman et al. (2019)). A large number of empirical analyses, on the other hand, render unknown structural weaknesses of a mechanism less likely. Thus, for critical use cases, algorithms are favorable that are both theoretically and empirically substantiated.

- **attribution to type of uncertainty** Better understanding what causes (unacceptably) high uncertainty estimates allows us to take appropriate countermeasures. Attribution is typically given w.r.t. either epistemic or aleatoric uncertainty, see related work in Section 6.1. The former corresponds to model-weight uncertainty and may be reduced by adding further input scenarios to the training data. The latter one, data-intrinsic uncertainty, is more difficult to handle and may require changes to the data collection process or the modeling task. For many real-world uncertainty use cases, clear distinctions between epistemic and aleatoric uncertainty are difficult since their training datasets are very sparse samples from high-dimensional input spaces.

Technical characteristics

- **applicability to large neural networks** Most real-world ML systems employ DNNs with millions of parameters. Applicability to, in that sense, large networks might thus be a desirable practical prerequisite for uncertainty estimation techniques.

- **minimal overhead** Calculating uncertainty estimates often requires additional numerical operations (e.g., for sampling-based approaches) or extensions to the neural model (e.g., for ensembling-based techniques). As many applications underlie resource constraints (especially mobile systems such as phones or drones) and more

generally for sustainability reasons, it is favorable to keep the "foot prints" of uncertainty estimators minimal w.r.t. storage and computing.

- **minimal trade-offs** While uncertainty quality and model performance are (in most cases) orthogonal requirements, there may be trade-offs between them in practice. This can be seen, for instance, on mechanisms that share weights to perform both task- and uncertainty-related predictions. It might therefore be desirable to use an uncertainty mechanism that leads to only minimal deteriorations in performance or other safety-relevant metrics.

- **technical simplicity** Being technically simple facilitates the integration of an uncertainty mechanism into neural architectures. Estimators, on the contrary, that require substantial changes to a network like additional layers or that rely on specifically annotated training data are harder to implement. They therefore incur a larger technical debt, which might make maintenance more challenging and increase the risk of programming errors.

Application-specific requirements

- Depending on the use case various additional requirements may exist. Semantic segmentation tasks, for instance, generate highly granular uncertainty outputs that render the following two requirements plausible: first, one may ask for coherent uncertainty estimation for the pixels belonging to a single object in an image, and second, one may expect pixel classification uncertainty to increase when approaching a boundary between semantic segments as such boundaries go along with irreducible data uncertainty. For the related computer vision task of object detection, it seems natural to ask for a positive correlation between estimated uncertainty and the degree of occlusion of a VRU. In the case of combined segmentation and object detection, one could require high segmentation uncertainties for image areas that contain undetected objects (false negatives).

This categorization scheme allows us to delineate the initial requirements more clearly, e.g., by splitting or merging them. It might moreover help to identify additional requirements that extend the initial set. While some of these requirement categories are qualitative, e.g., "being argumentatively substantiated", others are quantifiable, e.g., "being globally calibrated". The (relative) importance of the different requirement categories is moreover not fixed but varies for each uncertainty use case. In the following, we focus on the quantifiable uncertainty requirements and take steps to derive acceptance criteria from them.

6.4.2. Formalizing uncertainty acceptance criteria

Having disentangled the textual requirements using the categorization above, we further address those sub-parts that are specifically quantitative, e.g., issues of calibration. Concretely, we "translate" each quantitative requirement into an acceptance criterion

that consists of three key elements: first, a *semantic data specification* that reflects those regions of the input space that are relevant for a given requirement, second, *a measure of quality* that quantifies to which degree a requirement is fulfilled and, lastly, a *threshold value* that determines whether this degree of fulfillment is sufficient for the given DL system. While for some requirements, e.g., "minimal overhead", measures of quality are not uncertainty-related, e.g., run time, we focus on uncertainty-related aspects in the subsequent exposition. Please note that one uncertainty requirement may map onto several acceptance criteria.

Semantic data specification The semantic data specification of an acceptance criterion can be obtained based on the semantic dimensions that were used to describe the uncertainty ODD (see Subsection 6.3.2). For each of these semantic attributes, it is checked whether the uncertainty requirement renders it necessary to focus on certain value ranges and to exclude others. Requiring, for instance, that an autonomous car overtakes a cyclist only if it is highly certain that a safety distance of at least 1.5 meters can be kept at any point of this maneuver, might translate into a data specification that not only requires at least one cyclist per image but moreover a focus on short distances between cyclist and vehicle. The traffic participants aside, the road type of the input scenario is to be specified: while highways and country roads are not irrelevant, a clear focus for overtaking maneuvers are urban environments. In a subsequent step (see Section 6.6), the accordingly specified regions of the input space are mapped onto concrete datasets. In the following, we focus on the two other components of uncertainty acceptance criteria and provide best practices for choosing uncertainty measures (next subsection) and their respective threshold values (subsection after next).

Measures of uncertainty quality To determine to which degree requirements like global and local calibration are fulfilled and to quantify potential trade-offs between model performance and uncertainty estimation, scores of uncertainty quality are needed. Mathematically, these scores aim at concise descriptions of the mismatches between (often) high-dimensional probability distributions, mostly by means of scalar scores. These reductions of dimensionality bring about different sensitivities and distortions, compare, e.g., 2D map projections of a 3D globe where no map preserves areas, shapes and distances at the same time. Considering multiple uncertainty metrics, however, allows us to "construct" a more complete picture. This is of special relevance when optimizing a model on an uncertainty-related measure like negative log-likelihood (NLL, see Section 6.1). The resulting NLL-optimized model may neglect aspects of uncertainty that are not captured by NLL and alternative measures like ECE can be used to detect such overfitting to the optimization metric (in some fields also known as Goodhart's law (Chrystal et al., 2003)).

Aspects to differentiate between uncertainty measures are (among others) their sensitivity to underestimated and overestimated uncertainties as well as their sensitivity to outliers (in both directions). NLL, for instance, is unbounded and thus more sensitive to outliers compared to the bounded ECE. While this insensitivity of ECE to strongly deviating

uncertainty estimates limits its ability to resolve quality differences for a broad range of models, such outlier suppression might be acceptable for analyses that aim at measuring on-average uncertainty quality. Outliers aside, NLL is more sensitive to under-estimated uncertainties (asymptotically $\propto 1/\sigma^2$) compared to over-estimated ones ($\propto \log(\sigma)$). This implies that NLL-optimized models (on average) tend to overestimate uncertainties; a conservative behavior that might be considered advantageous from a safety perspective. ECE on the contrary treats over- and underestimated uncertainties roughly equivalent.

Taking a testing perspective, a pointwise measure like NLL has favorable properties compared to set-based measures like ECE. Search-based test (SBT) strategies explore the input data space to detect, e.g., input regions for which the model's uncertainty quality is low. For NLL, each new input data point provides a feedback on the (local) success of the search strategy as the point can be compared to its predecessors along the search trajectory. ECE in contrast requires a local data sample or the stored recent history of the search trajectory. However, NLL is not a calibration measure and thus not suited for the related requirement categories, namely, global and local calibration.

Both NLL and ECE capture on-average properties of uncertainty estimators as opposed to scores that "zoom in" and put emphasize on specific (safety-relevant) quantile ranges of uncertainty distributions. The expected tail loss (ETL, see Section 6.1), for instance, measures the depth of distributional tails.

Besides these generally applicable scores, task-specific uncertainty measures like probability-based detection quality (PDQ, Hall et al. (2020)) for object detection and area under the sparsification error curve (AUSE, Ilg et al. (2018)) for optical flow estimation exist. Reflecting the respective structures of inputs and outputs, they capture task-specific aspects of uncertainty and thus complement the generally applicable measures. For complex uncertainty use cases, one may additionally construct custom uncertainty measures, e.g., to involve prior knowledge about image segments or object types.

Finally, the standard versions of most uncertainty scores assume Gaussianity for model errors and uncertainty estimates, e.g., NLL and ECE. While these assumptions are (partially) justified for some modeling techniques or technical limits, empirical testing is called to analyze their validity for a given DL system.

Thresholding uncertainty measures To get from quantitative uncertainty measures to uncertainty tests that are typically formulated as binary decision rules, we require threshold values that indicate whether a test succeeds or fails. This binarization "transforms" statistical evaluations, which ask how good the quality of an estimator is, into semantic evaluations, asking whether the quality of an estimator is good enough for an application context. The following edge cases illustrate how strongly the notion of a semantically appropriate threshold value varies from use case to use case. For gambling, on the one hand, a win rate of 50.1% (for a binary game) may suffice to generate a profit *on average*, assuming the absence of fees and unlimited scalability. Critical ML applications, on the other hand, could require safe operations in *each individual situation*. In this case, even

well-functioning in 99.9% of all scenarios could indicate that in 1 out of 1000 situations a subsequent safety strategy should be considered.

Moreover, the already discussed challenges for breaking down high-level requirements become especially apparent for threshold values as these are typically formulated on a system level. To derive, for instance, a threshold value for an object detection (OD) network from a high-level requirement such as a maximum tolerable VRU collision rate, this threshold value, the collision rate, can be propagated along the AD model stack to the considered OD network, an inevitably qualitative process that requires various additional assumptions.

For a given DL system, an uncertainty metric may go along with several threshold values as the latter ones are set on the level of acceptance criteria and thus depend on their data specifications, e.g., whether the entire ODD, parts of it or transitions toward out-of-data are considered (see Section 6.6 for discussions on test datasets). Setting a threshold value for the negative log-likelihood (NLL) is especially challenging as it is a hybrid measure of network performance and uncertainty quality. A threshold value thus has to reflect both performance- and uncertainty-related requirements.

Results and remarks

The threshold values together with the above discussed uncertainty measures and semantic data specifications compose the quantitative acceptance criteria. Each quantitative requirement (from Section 6.3) should be formalized accordingly. Please recall that while uncertainty measures and threshold values are already fixed, the data specifications are still qualitative. Test datasets will be derived from them at the level of test cases in Section 6.7. These quantitative acceptance criteria, together with the qualitative ones, provide the "gold standard" for the next steps of the framework: first, the actual selection and adaptation of an uncertainty mechanism (step 3) and second, its quantitative testing (steps 4 and 5, compare respective arrows in Fig. 6.1).

6.5. Choosing an uncertainty mechanism

Comparisons between the desirable uncertainty properties outlined in the previous section and the actual properties of modern uncertainty modeling techniques allow us to choose suitable uncertainty estimators. To this end, we mirror the structure of Subsection 6.4.1 and analyze if and how well the 10 qualitative and quantitative requirement categories are addressed by representative uncertainty mechanisms.

Aspects of uncertainty quality

- **calibration** Self-assessing model performance appropriately requires to capture uncertainty comprehensively, i.e., both aleatoric and epistemic uncertainty. For transitions from ID to OOD, the importance of epistemic uncertainty (relative to aleatoric uncertainty) increases, rendering its modeling central for open-world

uncertainty use cases. While post-calibration routines, which often adjust a small number of global model parameters, help to improve uncertainty calibration in-distribution, this benefit does not necessarily carry over to out-of-distribution data (see, e.g., Snoek et al. (2019)). More advanced methods like MC dropout applied to networks that output the parameters of a Gaussian distribution (Kendall & Gal, 2017) and deep ensembles (Lakshminarayanan et al., 2017) improve on both ID and OOD calibration by combining parametric approaches that are optimized to capture ID aleatoric uncertainty with matching-based strategies for modeling epistemic uncertainty.

- **local calibration / heteroscedasticity** Aleatoric uncertainties are typically heteroscedastic, i.e., the data-intrinsic noise varies from data point to data point. While a homoscedastic modeling of heteroscedastic aleatoric uncertainty (like MC dropout) allows us to grasp the average level of data noise, it is more common to model it more flexibly, e.g., by means of network outputs that are interpreted as the parameters of a Gaussian distribution (as is done by an extension of MC dropout due to Kendall & Gal (2017)). Such parametric approaches aside, non-parametric approaches to model heteroscedastic aleatoric uncertainty exist, such as Wasserstein dropout (see Chapter 5) that reflect the data-point-dependent noise in the width of the network's output distribution that is generated by sampling sub-networks. While model-inherent (epistemic) uncertainty is in general also heteroscedastic, it typically increases, unlike aleatoric uncertainty, when leaving the training data distribution. The (implicit) ensembling methods outlined above model epistemic uncertainty via the spread of their sub-networks. On training data, these (sub-)networks are "matched", resulting in a small spread. Under data shifts, these matchings do not hold, the (sub-)networks disperse and thus yield larger uncertainty estimates.

- **ability to solve downstream tasks** While the evaluation of uncertainty estimators by means of downstream tasks is common in research, the exact tasks considered vary from method to method, rendering many results not readily comparable. Common types of tasks are the detections of misclassifications and outliers as well as distinctions between in- and out-of-distribution data. The latter task is specifically suited for uncertainty as it touches on the problem of uncertainty attribution (see respective category). Some methods like prior networks (Malinin & Gales, 2018) are especially suited to distinguish between different types of uncertainty and thus simplify downstream OOD detection. Practical limitations of this modeling approach were addressed by the more recently proposed posterior networks (Charpentier et al., 2020). In particular for high dimensions, comparisons between ID and OOD are often realized by evaluations on structurally different datasets (see, e.g., Snoek et al. (2019)). These analyses pose rather rough approximations of the continuous data shifts typically encountered in the real world.

Conceptual properties

- **argumentatively substantiated** Many uncertainty estimation techniques (e.g., stochastic gradient Langevin dynamics (Welling & Teh, 2011), MC dropout (Gal & Ghahramani, 2016a), deep evidential regression (Amini et al., 2020)) have their roots in Bayesian statistics and are, to varying degree, motivated in this regard, giving them a theoretical basis. Other techniques like deep ensembles take a frequentist perspective. Deep ensembles as well as MC dropout are considered to be among the most established uncertainty modeling techniques when judged by the high numbers of related publications and open-source implementations.

- **attribution to type of uncertainty** Attributing an uncertainty estimate to uncertainty sources requires a mechanism that models several types of uncertainty in the first place, especially aleatoric and epistemic uncertainty. As stressed in the related work, most such mechanisms internally employ combinations of "atomic" uncertainty mechanisms each of which modeling one type of uncertainty. Attributing uncertainty comes for such approaches "free of charge", e.g., for deep ensembles or MC dropout applied to networks that output the parameters of a learned Gaussian distribution. A fully parametric approach that provides a comparably easy decomposition is deep evidential regression. Prior networks for classification (Malinin & Gales, 2018) take uncertainty-source attribution one step further as they allow us to moreover distinguish distributional uncertainty from data and model uncertainty.

Technical characteristics

- **applicability to large neural networks** The methods discussed in the related work (Section 6.1) were pre-selected having a sufficient scaling behavior to be employed for application-size ML systems. Ensembling- and subsampling-based mechanisms, for instance, scale linearly with model size. For use cases employing small DNNs, the scope of practically applicable uncertainty mechanisms broadens as various Bayesian neural network (BNN) approximations (e.g., Liu et al. (2019)) become affordable.

- **minimal overhead** We analyze the overheads of uncertainty mechanisms w.r.t. numerical operations (processing time) and storage. Depending on the context, further dimensions may be considered such as the transmitted data volume in federated-learning applications (see, e.g., McMahan et al. (2017); Kamp et al. (2018)). Regarding numerical operations, parametric methods (Nix & Weigend, 1994) cause virtually no overhead as opposed to subsampling- and ensembling-based methods whose overhead is (typically) linear in the number of subsamples and model components, respectively. Similar considerations hold for networks like the probabilistic U-net (Kohl et al., 2018) that introduces an additional network branch and relies on sampling in a latent space. Efficiency-optimized versions of subsampling mechanisms such as last-layer variants (Snoek et al., 2019) and approximate analytical moment propagation (Postels et al., 2019) allow for noteworthy reductions of the

numerical operations and thus processing time. Regarding storage, (non-weight-sharing) ensemble methods (and approaches in that spirit that add, e.g., a network branch (Kohl et al., 2018)) require additional resources. Subsampling approaches (e.g., MC dropout or masksembles), i.e., in a sense weight-sharing ensembles, do not introduce any additional storage overhead.

- **minimal trade-offs** Uncertainty mechanisms impact network performance, in most cases, by modifying either the network capacity or its optimization objective. A parametric model for a regression task, for instance, does the latter as it optimizes Gaussian likelihoods (NLL) instead of squared errors (RMSE). This simultaneous optimization of mean and variance typically causes a certain decrease of performance (at least on training data). On the contrary, both subsampling- and ensembling-based methods change the network capacity, however, in opposite directions: subsampling limits the capacity as for each forward pass weights are omitted while ensembling increases the capacity as multiple models are learned for the same task. Trade-offs (or synergies) between uncertainty estimators and other trustworthy-ML components can be expected (see Section 6.1), but are difficult to quantify due to the large variety of the latter ones.

- **technical simplicity** Being technically simple is a driving factor for the wide adoption of an uncertainty mechanism. The modifications on code level induced by standard approaches such as subsampling-based, ensembling-based, and parametric ones are in most cases minimal. For the mentioned approaches, the network is slightly modified (subsampling), a loop over model training is added (ensembling) and the objective function is modified (parametric). Examples for more involved techniques are SWAG (Maddox et al., 2019) and probabilistic U-net (Kohl et al., 2018) that aggregate statistics along the training trajectory and require an additional network branch, respectively.

Application-specific ones

- The main types of uncertainty estimators are flexible enough to be employed in a variety of neural architectures and allow us to estimate uncertainties at different granularity levels. Of special importance is low-level uncertainty information, e.g., on pixel level, as it enables us to construct custom uncertainty aggregates for ML applications that require non-standard uncertainty "entities". Approaches like probabilistic U-net (see ibid.) and stochastic segmentation net (Monteiro et al., 2020) help to strengthen the coherence between such pixel-level uncertainty estimates. To further improve the susceptibility of a segmentation model w.r.t. critical segments such as VRUs, one may weight the segmentation outputs with pixel-resolved prior class probabilities (see, e.g., Chan et al. (2019)).

Many state-of-the-art uncertainty estimators combine different types of mechanisms to mitigate conceptual weaknesses and to leverage their respective strengths, e.g., to describe both aleatoric and epistemic uncertainty. Complex use cases may require an

adaption of existing mechanism combinations or even the construction of new ones. One may further analyze whether an uncertainty mechanism introduces (steerable) trade-off relations between some of the 10 requirement categories.

Results and remarks

Having picked an uncertainty mechanism, it needs to be optimized. Some mechanisms render post-training of the ML model or even re-training from scratch necessary, for others a less involved post-processing routine suffices (see, e.g., temperature scaling (Guo et al., 2017)). The "original" network without uncertainty estimator may be kept for comparisons, e.g., to detect side effects of integrating the uncertainty mechanism such as potentially occurring reduced task performance.

6.6. Scope and structure of uncertainty testing

Given an optimized uncertainty estimator, we are headed to test whether it fulfills the uncertainty acceptance criteria (see Section 6.4). Before we construct concrete test cases in Section 6.7, we introduce a test hierarchy that builds on semantic and non-semantic test data selection strategies and discuss how it aids in the testing of the uncertainty acceptance criteria. We focus on quantitative testing as qualitative acceptance criteria (e.g., being theoretically justifiable or allowing for uncertainty attribution) were (in most cases) already addressed when choosing an uncertainty mechanism in the previous section. The goal of this step in the framework (recall Fig. 6.1) is to determine the test depth and the test focuses for each quantitative acceptance criterion.

Different from testing programmed software, approaches that cut a DNN in smaller pieces are challenging due to the highly interacting nonlinear (black-box) nature of these models, also referred to as "changing anything changes everything" (CACE, see Sculley et al. (2015)). We therefore consider the entire DNN as a single unit for testing. Thus, test hierarchies from classical software testing (from unit over integration to system tests) are not easily transferable (see, e.g., Gannamaneni et al. (2021)). The complexity of (real-world) ODDs moreover renders comprehensive testing of (uncertainty) acceptance criteria difficult (see outline in the beginning of the chapter). Diverse testing types and testing methods, however, yield a broad overview of the strengths and weaknesses of an uncertainty estimator within a given computing budget.

We propose a test hierarchy that builds on iteratively refined test data selection strategies, as it is mostly not possible to conclusively map the ODD (or representative parts of it) onto a dataset. The data selection hierarchy is a practical vehicle serving a twofold purpose: on the one hand, it allows us to broadly cover the entire ODD and, on the other hand, to conduct deep-dive analyses exploring challenging and (safety-)critical regions of it. Specifically, we distinguish four hierarchy levels:

- *Technical tests* (Subsection 6.6.1) that check elementary technical properties of an uncertainty estimator. The semantics of the employed test inputs are (largely) irrelevant.
- *Global uncertainty tests* (Subsection 6.6.2) that focus on global properties of an uncertainty estimator such as its on-average quality within and outside the performance ODD.
- *Subset and pointwise uncertainty tests* (Subsection 6.6.3) analyze worst-case behaviors (and other quantile ranges) of the uncertainty estimate distribution and how well an uncertainty estimator functions for individual (application-critical) input scenarios.
- *Complementary uncertainty tests* (Subsection 6.6.4) are an open residual test set to explore more involved uncertainty properties as well as novel testing methods.

For a schematic visualization of the test data selection concepts on the four hierarchy levels, see Fig. 6.4. Before introducing the test hierarchy in detail, we stress that it is applied to each (quantitative) uncertainty acceptance criterion separately: first, we determine the *test depth* for the criterion, i.e., the levels of the test hierarchy to be considered for adequately addressing it. Each of these test hierarchy levels in turn contains several test types, each of which putting a focus on another relevant characteristic of uncertainty estimates. These test types are finally broken down onto concrete test cases (see Section 6.7).[3]

6.6.1. Technical tests

A first step toward thoroughly examined uncertainty estimates is testing their basic technical properties. We begin by checking the interfaces for uncertainty information, especially whether they comply with the flows and granularities determined in Subsection 6.3.3. The data types of the interfaces are relevant as they set the ranges and resolutions of the uncertainty estimates. Given the interfaces, we test how an estimator handles undefined or invalid inputs, e.g., a corrupted negative uncertainty estimate from a previous ML model in a model chain and whether explicit sanity checks for uncertainty outputs are implemented, e.g., whether a flag is raised for negative or close-to-zero estimates. For black-box ML models with access constraints, these output-related checks cannot be conducted as they require knowledge about the model's inner workings. Some uncertainty mechanisms (applicable to white-box networks) go along with additional model parameters (e.g., ensembling methods, see Subsection 6.3.3). Such mechanisms call for analyses of whether these additional parameters were properly adjusted by network (re-)training, i.e., if their values are, e.g., continuously distributed or that none of them is undefined.

Running an uncertainty mechanism on a larger test dataset allows us to examine its processing time distribution that provides insights on average processing times (especially relative to the processing times of the same DNN without an uncertainty estimator) and

[3] While out of scope for this thesis, the presented data-driven black-box testing approach is model-agnostic and could thus be evaluated by using it for a model that can be tested in an additional way. This is typically true for white-box systems where components can be investigated individually.

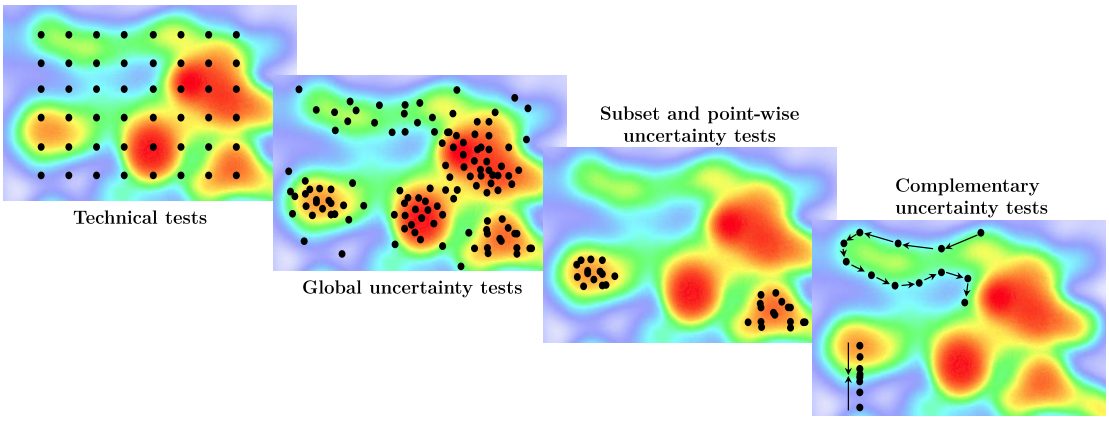


Fig. 6.4.: Schematic visualization of the data selection concepts on the different levels of the test hierarchy. The heatmap shows a fictitious data density (violet is low, red is high). Simple technical tests (left panel) are based on a coarse data selection. Global uncertainty tests in contrast (second panel), rely on a broad, density-appropriate sampling of the input space. Subsequent subset and pointwise uncertainty testing (third panel) focuses on particular input scenarios and regions. Complementary uncertainty tests (right panel) finally evaluate more complex properties of uncertainty estimates. The actually conducted tests strongly depend on the uncertainty use case, see the following subsections for details.

potentially occurring latency "tails" that need to be handled. Finally, one may require uncertainty estimators to be deterministic for test reproducibility. As many uncertainty mechanisms rely on sampling routines, this requirement translates into the question of whether all employed pseudo-number generators are seeded.

6.6.2. Global uncertainty tests

Recalling Subsection 6.3.2, the uncertainty ODD is, in most cases, intended to be a superset of the DNN's performance ODD, i.e., the uncertainty estimator is expected to reliably predict the model's confidence on inputs from both within (ID) and outside (OOD) the model's training distribution. Therefore, global uncertainty tests investigate the general functioning of an uncertainty estimator on the broader uncertainty ODD and are conducted on accordingly compiled datasets. Intermediate test results per data point are (typically) aggregated to provide dataset-wide uncertainty quality figures.

For testing within the performance ODD, one may build, if available, on the datasets used for the performance evaluation of the network or otherwise on structurally similar datasets.[4]

[4]In general, we assume that both network and uncertainty estimator are (concurrently) trained on the same dataset. Assuming further that the uncertainty estimator captures epistemic uncertainty, we expect its "domain of proper functioning" to be larger compared to the one of network performance.

Possibly occurring performance degradation due to the inclusion of an uncertainty estimator (performance-uncertainty trade-off) should be investigated. While further insights on performance may arise during the global tests, they are not the primary focus.

Next, we examine the uncertainty estimator's quality in the "outer" areas of the uncertainty ODD, i.e., on data structurally different from the training data. Such data shifts introduce additional epistemic uncertainty and can be characterized by the types of concepts they change and the degree to which concepts are changed, respectively. An example are the discrete data shifts that occur when model and uncertainty mechanism are trained on simulated inputs (as virtual open worlds allow, e.g., to generate arbitrarily many critical scenarios) and evaluated on real-world data. The (simulation-based) StreetHazards anomaly segmentation dataset (Hendrycks et al. (2019)) on the other side introduces high-level shifts as it modifies the semantics of traffic scenes by placing unknown unknowns (random objects like sheep and airplanes) into them. For a given uncertainty use case, the data specifications of, e.g., the uncertainty ODD (see Subsection 6.3.2) and the acceptance criteria (see Subsection 6.4.1) help to select relevant data shifts. Low-level distortions like noising, blurring or changed colors and contrasts are of interest for most uncertainty estimators as they imitate flawed measurements and (likely) increase the aleatoric uncertainty of the input.

For ML models that process, for instance temporal, input streams, we test the (temporal) consistency of uncertainty estimates, i.e., whether the change rates of uncertainty estimates reflect the input change rates.[5] This becomes especially important when (simple) algorithms for temporal smoothing like Kalman filters (Welch & Bishop, 1995) are employed that may limit uncertainty change rates.

6.6.3. Subset and pointwise uncertainty tests

An uncertainty estimator that is insufficient w.r.t. the above tests of global functioning is likely to be discarded. Successfully passing those tests, however, is not a sufficient but only a necessary condition as they solely check aggregated uncertainty quality. Many uncertainty use cases, in contrast, require proper functioning in individual situations, especially in (potentially) critical ones.

One way to study such critical inputs is revisiting the global tests, more concretely, their pointwise intermediate results (e.g., normalized residuals or NLL values). Instead of averaging them (e.g., by means of ECE or mean NLL), one may stay at a level of higher granularity and focus on individual quantiles or quantile ranges of these distributions. For details on such *output-uncertainty-based testing*, see the paragraphs below.

Datasets can not only be sliced based on numerical uncertainty scores but moreover based on high-level annotations, assuming that such information is available for each input data point. Accordingly derived *semantic-dimension-based tests* (see below) reveal high-level strengths and weaknesses of an uncertainty mechanism and moreover allow us

[5] If scenes are, for instance, only very slowly changing, one would expect the uncertainty estimates to decrease over time while sudden changes in a scene would expectably lead to an increase in uncertainty.

Tab. 6.1.: Four technical approaches to subset and pointwise uncertainty testing that are categorized based on data selection strategy (x-axis) and their (non-)semantic nature (y-axis), respectively. See Subsection 6.6.3 for details.

subset and pointwise uncertainty testing	**distributional**	**search-based / curated**
non-semantic	output-uncertainty-based testing	testing with uncertainty-based search strategies
semantic	testing along semantic dimensions	testing critical semantic input scenarios

to evaluate how well its domain of proper functioning covers the desired uncertainty ODD as specified in Subsection 6.3.2.

Going one step further, one may not only slice given datasets but curate (or even create) potentially critical inputs based on semantics. The sampling-based approach of determining the uncertainty ODD in Subsection 6.3.2 guides these data compilations that provide the basis for *testing uncertainty properties in critical semantic scenarios* (see below).

Moreover, critical input scenarios may be generated by following hints on low local uncertainty quality. Such *uncertainty-based search strategies* (see below) allow us to explore the input space to uncover regions of insufficient self-assessment. Given the sparsity of most datasets and the incomplete nature of semantic specifications, this test type poses a promising technical approach to reduce blind spots of subset and pointwise uncertainty testing.

In the following paragraphs, we discuss these four testing techniques in detail (see Tab. 6.1). Fig. 6.5 illustrates the differences between the sketched testing strategies by means of a symbolic 2D input space with a semantic and a non-semantic axis.

Output-uncertainty-based testing Slicing a dataset based on local output uncertainty quality requires a pointwise uncertainty measure, rendering set-based (calibration) metrics like ECE unsuited. More appropriate candidates are absolute values of uncertainty estimates and measures that put pointwise uncertainty estimates and (absolute) model errors into relation, like NLL values or normalized residuals. The choice of the measure is guided by the uncertainty acceptance criteria (see Section 6.4).

Calculating the width of an uncertainty-output distribution (its variance or, for multi-dimensional output distributions, its covariance) provides insights on the meaningfulness of on-average uncertainty figures for individual input scenarios as these "width" scores measure the (average) deviation from mean behavior. Going beyond average deviations,

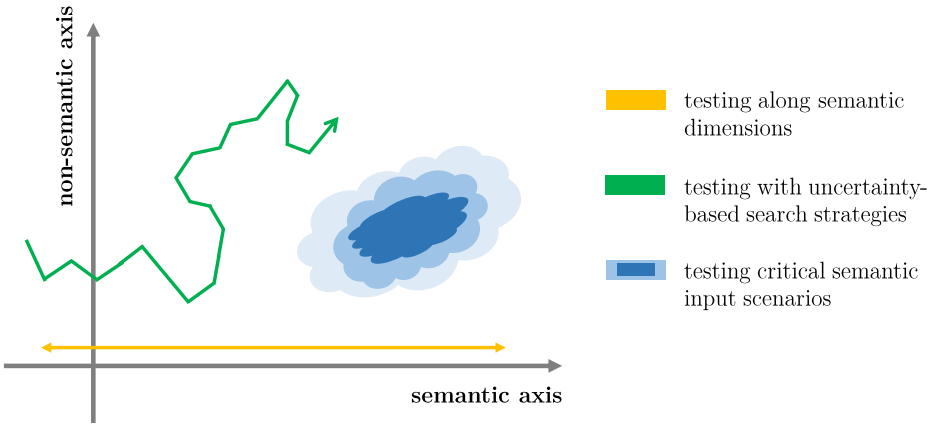


Fig. 6.5.: Symbolic illustration of three out of four data selection strategies on the testing hierarchy level of subset and pointwise tests. The data selection for output-uncertainty-based testing is trivial as it builds on the standard test dataset used for global testing (see above) and is thus not shown. The high-dimensional input space is sketched as a 2D plane with a semantic and a non-semantic axis. Testing along a semantic dimension (orange arrow) requires (in most cases) a simulation environment that allows us to systematically vary the corresponding input attribute while keeping all other input properties unchanged. However, in many real-world use cases, relevant regions of the input space (blue "cloud") are too complex to fully capture them this way. Instead, they can be represented, e.g., by expert-curated datasets. Search strategies finally do not require semantic information and may, for instance, be driven by gradients of local uncertainty quality (green trajectory). For details on all data selection strategies, see Subsection 6.6.3.

we next focus on sub-datasets that cause strongly diverging uncertainty estimates and thus contribute to the tails of the uncertainty-output distribution. For pointwise measures like NLL and normalized residuals, these tails correspond to uncertainty estimates that severely under- or overestimate the actual model errors. As discussed above, one may argue for the special practical relevance of under-estimated uncertainties, i.e., of overconfident model predictions. These worst-case scenarios w.r.t. uncertainty quality are measured by quantile values (e.g., 1% or 99%) or measures that are sensitive to the depth of the distributional tails, such as conditioned mean values (e.g., the ETL). Since these measures allow us to detect deviations from Gaussianity (see, e.g., Section 4.2 for an analysis of these deviations), they may help to identify (overly) simplifying modeling assumptions and thus provide an additional safeguard.

For ML models processing structured input, it may be insightful to attribute output uncertainty to input features. Input modifications like obfuscations allow us to analyze, on the one hand, the sensitivity of output uncertainties w.r.t. input regions (such as image patches, frequency bands or sub-graphs) and, on the other hand, how susceptible

an uncertainty estimator is w.r.t. local and global input features (e.g., whether or how strongly an object-level uncertainty changes when a far-off image patch is modified).

Relying on output uncertainty estimates, these tests do not require (high-level) meta-information. Being available, however, such semantic information may allow for an efficient and human-understandable description of the selected sub-datasets. In contrast, the availability of high-level meta-information is a prerequisite for testing along semantic dimensions (see the next paragraph).

Testing along semantic dimensions Real-world ML systems likely encounter inputs at inference that are statistically novel compositions of known concepts (e.g., of known object types) and moreover structurally new concepts (unknown unknowns). Analyzing how uncertainty quality changes in the presence of increased epistemic uncertainty due to varied scene semantics provides a high-level understanding of the estimator's extrapolation capabilities and its borders of functioning. Concretely, one may study an ID-OOD transition along a semantic "direction",[6] e.g., whether object-level uncertainty estimates for VRUs are sufficiently calibrated as a function of their distance to the vehicle and how calibration quality may decrease for far away (and thus "small") VRUs. Another example of a semantic dimension in the field of AD are (labeled) body parts of VRUs that allow us to analyze whether uncertainty quality diminishes for various kinds and degrees of VRU occlusion (another ID-OOD transition).

Such tests enable us to compare the actual abilities of an uncertainty estimator with its ODD specification that may, e.g., ask for properly calibrated uncertainty estimates for VRUs that are within the breaking distance of a vehicle. Technically, such analyses are based on either richly annotated real-world data or simulated data. While the latter ones allow for a systematic variation along a semantic dimension as well as for "zooming" into critical sections (as arbitrary amounts of data can be generated), artificial data comes at the cost of a domain gap between, on the one hand, the accordingly conducted tests and, on the other hand, the actual usage of the ML model in the real world.

Moreover, these variations along only one semantic "axis" at a time constrain the variability of the reachable inputs (one may think of a basis scene that is, one after the other, deflected in various directions). While these semantic-axis tests provide a first understanding of high-level properties of an uncertainty estimator, further testing on more realistic input scenarios is required.

Testing critical semantic input scenarios Critical input scenarios are barely represented in standard datasets. Moreover, they are often too complex to encounter them when varying a single semantic dimension of a basic input scenario using, e.g., a simulation engine (see previous paragraphs). Both aspects, their rare occurrence and their complexity, render a decrease in model performance likely and thus large uncertainty estimates

[6] We consider only modifications along one semantic "axis" at a time to avoid combinatorial explosions. Approximate techniques that study the impact of combinations of semantic concepts exist, e.g., pairwise testing (see Kuhn et al. (2004)).

desirable. Datasets of such critical input scenarios (Chan et al., 2016; Yao et al., 2019; Bao et al., 2020) can either by curated by domain and safety experts (e.g., Jermakian & Zuby (2011)) or semi-automatically generated using, e.g., probabilistic scene grammars (Kar et al., 2019) that set up scenes based on pre-defined probability distributions.[7] The sampling-based approach to determining the ODD from Subsection 6.3.2 provides a notion of what constitutes a critical scene for an uncertainty use case and is thus a natural starting point for data curation or data modeling. Also, critical historical scenarios can be included if available (e.g., data of historical human-induced crashes in the field of AD (Scanlon et al., 2021)).

For ML models processing spatio-temporal data streams, one may moreover consider critical scenarios in the temporal domain such as abrupt changes of lighting conditions in the case of AD, e.g., when driving in a tunnel or when the sun breaks through the clouds, and test whether uncertainty estimates change accordingly.

Testing with uncertainty-based search strategies Available datasets typically capture only a small subset of the input space and thus reveal only a potentially small fraction of the actual weaknesses of an uncertainty estimator. To enlarge the data base, one could synthesize additional input points, e.g., by interpolating between or extrapolating from given test data points. More systematic, however, compared to such untargeted data acquisition strategies, are "pro-active" searches for input space regions that cause, e.g., large or ill-calibrated uncertainty estimates.

Various algorithmic search strategies to explore input data spaces exist (e.g., Pei et al. (2017); Odena et al. (2019); Klischat & Althoff (2019)). They (typically) follow hints of weak model performance, e.g., by means of gradient descent in a latent space. Adversarial attacks fall into this category as well as coverage-based strategies (see related work in Section 6.1). While most such approaches focus on model performance, we advocate for search strategies that are guided by uncertainty quality. They require pointwise uncertainty measures such as NLLs or normalized residuals and target larger regions of input space compared to performance testing due to the (in most cases) larger uncertainty ODD. Depending on the type of uncertainty estimator, detected large uncertainty estimates can be attributed to either aleatoric or epistemic uncertainty.

Relying on numerical (pointwise) uncertainty scores, search-based tests (SBTs) do not require any meta-information, unlike testing along semantic dimensions and testing of critical input scenarios. However, non-semantic SBTs may still benefit from high-level annotations as the latter ones allow for (approximate) semantic descriptions of detected weaknesses (compare output-uncertainty-based testing).

6.6.4. Complementary uncertainty tests

The discussed testing approaches (see Tab. 6.1) are by no means definitive and further (especially application-specific) tests are required. The complementary uncertainty tests

[7] The parameter distributions are either directly or indirectly (when learned from curated data) set by experts.

are intended as an open residual set where as one example we present "cross-analysis" examinations. We outline two variants of such tests: firstly, combinations of the uncertainty testing techniques discussed above, e.g., quantile range uncertainty analyses on a curated dataset of critical input scenarios. Secondly, "cross-analysis" tests in the sense that uncertainty estimates (and not the predictions of the model) are analyzed w.r.t. other safe-ML dimensions, such as interpretability or fairness. Considering, e.g., the fairness of uncertainty estimates, one may test whether uncertainty quality is comparably good across the values of sensitive input attributes (e.g., gender, color of skin or religious clothing in the case of a VRU detector).

Results and remarks

The above presented testing hierarchy facilitates structured testing of uncertainty estimators where the sketched testing methodologies help to cover a broad range of relevant input space regions, paying attention to both their semantic and non-semantic aspects. More concretely, we choose an appropriate test depth (in the test hierarchy) for each quantitative uncertainty acceptance criterion. For the respectively selected hierarchy levels relevant focus points of testing can be determined, e.g., testing of low-level sensory distortions (on the hierarchy level of global tests) and testing along semantic dimensions (on the hierarchy level of subset and pointwise tests).

6.7. Instantiating, executing, and evaluating uncertainty test cases

To enable statistical testing, the identified focus points of uncertainty testing (see previous section) need to be mapped onto concrete uncertainty test cases. Best practices for formulating such test cases are outlined in Subsection 6.7.1. Aspects of running the specified uncertainty tests are sketched in Subsection 6.7.2. Given the (binary) results of the test cases, we provide guidelines on how to derive an overall evaluation of an uncertainty estimator w.r.t. the uncertainty acceptance criteria in Subsection 6.7.3.

6.7.1. Instantiating uncertainty test cases

Uncertainty test cases can be understood as instantiations of high-level testing focuses (see Section 6.6). Similarly to the derivation of uncertainty acceptance criteria from qualitative requirements, these uncertainty tests are best formulated redundantly and collaboratively, i.e., involving both domain and ML experts, to reduce blind spots. Each testing focus is typically addressed by several test cases. Guided by the uncertainty acceptance criteria, the data bases, threshold values[8] and technical configurations (e.g., the meta-parameters) for these test cases are determined. Such specifications may vary between test cases, even

[8] While the threshold value is in general given by the underlying acceptance criterion, it might be refined for some test cases, especially for subset-based ones. When investigating, e.g., worst-case scenarios of a given dataset, average values no longer hold.

between those that address the same acceptance criterion. Uncertainty tests building on search strategies generate the test dataset during test execution and thus do not require fully pre-specified datasets. Instead, their meta-parameters need to be set.

A prerequisite for a test to be conceptually meaningful, is its statistical significance that involves, among others, the following two aspects: first, whether a test dataset contains a broad set of relevant concepts (for the use and test case at hand) such that evidence-based conclusions can be drawn regarding the quality of the ML model at inference. Second, whether the outcome of the test critically depends on low-level technical parameters such as random seeds for initialization. Insights in this regard may be gained by repeatedly executing the test in varying configurations and analyzing the sensitivities of the according test outcomes. Apart from statistical considerations, one may estimate the "foot print" of a test w.r.t. computation or storage, i.e., the resources its preparation and execution consume. While the result in absolute terms may determine whether a test is affordable, relative results are helpful for test ordering and prioritization.

Comparing two test cases, each of their components (test dataset, uncertainty measure, threshold value, technical specification) might introduce overlaps or discrepancies between them. While certain overlaps are desirable (see above), we raise awareness for less obvious (implicit) dependencies between test cases: These range from test data bases that have superset-subset relations or are strongly overlapping over measures of uncertainty quality that are correlated to threshold values that might be at odds. Finally, one may analyze whether the "union" of all test cases sufficiently addresses the chosen focus points of testing or whether unaddressed blind spots remain.

6.7.2. Running uncertainty test cases

Since the uncertainty test cases are constructed to be (largely) independent of one another, they may be executed in an arbitrary order. Following the structure of the test hierarchy, from general technical tests to specific complementary tests, may however be beneficial as potentially occurring severe weaknesses are detected at an early stage (in the worst case, fail fast). Having executed the uncertainty tests, their outcomes are uncertainty values that either exceed or fall below the threshold value of the respective test thus providing a binary test result ("passed" or "failed"). In the case of conflicting results for conceptually similar tests, it may be worthwhile to investigate whether the diverging outcomes are, at least to some extent, attributable to a specific difference in their test setups, e.g., to varying sensitivities of uncertainty measures. Such insights may result in better designed or additional test cases (compare respective backward arrow in Fig. 6.1).

6.7.3. Evaluating uncertainty test cases

Addressing uncertainty acceptance criteria by means of test focuses (organized in test levels) that are in turn mapped onto a set of test cases, introduces hierarchical tree-like structures. Once the (final) set of tests is executed and all binary test results are available, these information at the "leaves" of the logical trees must be aggregated to obtain an

overall evaluation of an uncertainty mechanism w.r.t. the acceptance criteria (the roots of the logical trees). Aggregating the binary uncertainty test results and, in a second step, the uncertainty acceptance criteria requires a qualitative, highly application-specific argumentation that could be formulated using, e.g., a goal structuring notation (Kelly & Weaver, 2004). Depending on the uncertainty use case and the "level" of the acceptance criteria, such argumentations may be strict, i.e., all acceptance criteria must be fulfilled (and maybe even all tests must be passed successfully) or more flexible, i.e., a custom argumentation (employing, e.g., weightings of tests and a custom "merging" logic) allows for a positive overall evaluation despite a few non-fulfilled acceptance criteria (or at least despite a few failed tests).

Results and remarks

Insights during testing and test result aggregation may trigger (iterative) adaptations of its scope and structure. In the case that a "converged" testing strategy, i.e., one leading to consistent test outcomes, results in an overall negative evaluation of an uncertainty estimator, uncertainty modeling may start all over again (compare respective backward arrows in Fig. 6.1). If, however, testing leads to an overall positive evaluation of an estimator w.r.t. the uncertainty acceptance criteria, the end point of the proposed development and testing framework is reached. In the case that high-level requirements change at a later date, e.g. due to external influences such as novel technical regulations, it may be necessary to re-apply the uncertainty framework. In these situations, its modularity allows for the re-use of results from the previously conducted development and testing steps.

6.8. Discussion

The importance of uncertainty estimation to establish safe ML systems is widely agreed. While various technical uncertainty mechanisms were put forward, less light is shed on how to systematically address application demands with them. We propose a framework that approaches uncertainty quantification from a practitioners' perspective, i.e., starting from the various high-level requirements an uncertainty estimator is subject to. These range from the underlying use case that predetermines its desired properties and operational design domain to technical specifications that may limit its depth of integration into the ML model. For a more systematic analysis, each requirement is grouped into one of ten requirement categories. Formalizing the requirements as acceptance criteria with semantic data specification, uncertainty measure and threshold value yields the "gold standard" for the subsequent modeling and testing steps. While a custom uncertainty modeling technique may be constructed to fulfill these uncertainty acceptance criteria, it is often worthwhile to use established uncertainty modeling techniques as building blocks. We guide this construction of a *tailored* estimator by matching the uncertainty requirement categories with central technical properties of uncertainty modeling (e.g., modeled uncertainty types, generalization ability, attribution to uncertainty source). The resulting requirements-informed uncertainty mechanism is systematically tested to uncover (potentially existing)

structural weaknesses that might hinder its further use or deployment. Technically, this is achieved by combinations of semantic and non-semantic testing approaches that are organized in a test hierarchy. For each acceptance criterion, a test depth (according to the hierarchy) and test focuses are determined. Instantiating these test focuses as concrete test cases allows us to finally obtain (binary) test results. An overall evaluation of these results yields an answer whether the uncertainty estimator meets the uncertainty acceptance criteria, and thus, whether it is suitable for the given DL system. These individual steps within the framework have well-defined hand-over points that allow for a strong degree of encapsulation. This can be beneficial for larger projects where multiple mechanisms are being developed concurrently or when comparable testing approaches are needed for multiple products.

Furthermore, several steps of our conceptual framework can be operationalized and automatized in software frameworks. For instance, the selection of uncertainty models and metrics, as is done in existing frameworks like *UQ360* (Ghosh et al., 2021) and *uncertainty-toolbox* (Chung et al., 2021a). Testing and artifact management may be made traceable and reproducible using ML lifecycle platforms like *MLflow*. Desirable enhancements of such tools might moreover contain components such as requirement templates, building blocks for qualitative (safety) argumentations and backlogs of standard tests. Particularly, the (semi-)automation and optimization of quantitative testing bears potential. Promising technical approaches in this regard range from ML-based strategies to speed-up a concrete test case (e.g., by predicting, right after starting the test, whether its outcome will be negative), over the (e.g., Bayesian) generation of the best follow-up test configuration for a pre-defined test type to the development of heuristics on how to effectively react on failing test cases. Such software frameworks may moreover enable effective test-driven design and development of uncertainty estimators. Gained insights and uncovered weaknesses could even fuel research on novel uncertainty modeling techniques.

While our framework structures the testing of a single model, industry applications often have a multi-step development approach. It might be worthwhile to investigate to which extend early, and therefore easier to obtain, test results are indicative for success or failure of later stages of development. The proposed encapsulation of testing from development might help in the necessary transfer of results. Achieving a strong correlation, for instance with respect to qualitative uncertainty properties, may help to streamline the development process or to identify critical conceptual building blocks of the DL system.

Technical requirements aside, various official regulations for learned systems are about to emerge, especially in the EU (European Commission, 2021), where, likely from 2025 on, newly created and substantially changed DL applications will be regulated by an "AI Act" if the respective application type is considered to be critical. Our framework may be of help for associated product and process examinations, especially those concerning model reliability and the handling of unknown inputs. It might also serve as a structural aid for the technical documentations required by the prospective EU regulations.

While uncertainty estimates help to fulfill requirements for trustworthy ML, there are multiple other ways to either increase reliability, e.g., robustified training (Madry et al.,

2018), or other aspects of trustworthiness, for instance fairness or interpretability. The proposed framework, despite its specific focus on uncertainty quantification, remains applicable in these cases. The main difficulties it addresses, namely, the black-box nature of the neural models and their high-dimensional input spaces, are common to most safe-ML techniques and approaches. The proposed hierarchical test strategy, for instance, is transferable, as it addresses the high-dimensional input spaces by specific combinations of (semantic and non-semantic) testing methodologies and test data selection routines. Moreover, the ways in which the underlying use case and technical specifications predetermine the conception and implementation of the safe-ML techniques carry over to other dimensions of safe ML. At the level of technical properties, however, some requirement categories like "calibration" or "attribution to type of uncertainty" as well as mechanisms and measures are uncertainty-specific and need to be replaced by appropriate (statistical) concepts for the respective dimension of safe ML (e.g., by "interpretable-by-design" for qualitative interpretability requirements or "group and individual fairness" as a quantitative demand for fair ML models).

Such procedures for developing and testing safe-ML tools may be complemented, especially in larger organizations, by role-based access and rights management that builds on the encapsulation of our approach and enhances the interfaces between the various steps and the teams involved (e.g., development and product). Distributing responsibilities in that way seems especially advantageous for the development and testing of ML components: development teams, for instance, may not have access to the concrete test cases to avoid model optimization that overfits to these cases. Alternatively, a part of the test cases may be hidden from the developers, while others are still accessible to them. This may be compared to the public and private datasets of ML challenges. Taking inspiration from IT security (Roy et al., 2010; Zhang & Zhuang, 2019), testing teams could moreover act as attackers that continuously challenge the uncertainty estimators developed by the modeling team, a competitive serious game that may result in more robust ML systems.

In many real-world applications, the deployed DL systems are not static, but further improve during operations, either due to continuous learning or, and more likely for most safety-critical applications, after scheduled updates that are reviewed before going into production. For such an updated model, the question arises whether testing and validation need to be done all over again or whether existing test results from previous versions of the model can be re-used. In particular, an interesting aspect could be to determine whether mild deviations in high-level tests of the hierarchy can yield insights on the transferability of previous results on the deeper parts of the hierarchy. Should this hold, the testing hierarchy may contribute to more efficient model validation in the long run.

By proposing the presented framework, we contribute a holistic, application-driven perspective on uncertainty estimation in DL models. Our guidelines may assist in a more appropriate choice or development of uncertainty estimators, allowing for a use of this promising modeling technique that is tailored to match application demands. The multi-staged testing procedure not only helps to satisfy upcoming ML regulations but also increases the overall reliability, and thus the quality, of the DL systems.

7. Conclusion

A promising and feasible approach toward trustworthy ML applications is to establish a "safety net" of interlinked technical and non-technical measures. Threads of this "net" are i) basic learning and estimation mechanisms that are thoroughly understood, ii) ML methods constructed on top of them that leverage the strengths of these building blocks while mitigating their weaknesses, and iii) ML systems that are designed and evaluated in ways that reflect use case requirements in each step of their development. In this work, we contribute (as detailed below in "Summary") to weaving this "safety net" with a focus on uncertainty estimation. Techniques to quantify uncertainty form an important sub-field of trustworthy ML, as they explicitly acknowledge limitations of prediction quality due to model size constraints and finite, ambiguous datasets. Uncertainty quantification can thus be regarded as a self-assessment—as the meta-task of estimating a model's lack of knowledge about the "true" answer.

Summary Regarding the "threads" mentioned above, we contribute, firstly, by investigating how uncertainty is affected by model capacity, and, in this way, more thoroughly understand the uncertainty modeling abilities of MC dropout. Based on these insights, we construct, secondly, Wasserstein dropout, which extends the dropout mechanism by more strongly diversifying ensemble members. Thus, it additionally captures heteroscedastic data uncertainty. Finally, we structure and address use case-specific requirements on uncertainty estimation. More concretely, we provide a hierarchical testing methodology based on data-driven test cases to examine not only global uncertainty properties but, more fine-grained, uncertainty quality on critical data "slices" and in individual scenarios.

Within the first part of this work (Chapter 4), we investigate how capacity constraints affect uncertainty modeling, specifically, the properties of predictive uncertainty distributions (see Section 4.2) and the quality of modeled uncertain dynamics (see Section 4.1). On a technical level, we influence and steer model complexity by varying the (actual or effective) size of a model's hidden parameter space.

For models with discrete parameter spaces, namely, hidden Markov models (Rabiner & Juang, 1986), we analyze the transition between high-quality and low-quality modeling regimes and demonstrate the benefits of a novel word2vec-inspired (Mikolov et al., 2013b) reparameterization that facilitates the close approximation of stochastic data dynamics while requiring significantly fewer parameters compared to the standard parameterization.

The properties of uncertain outputs are studied in more detail for neural models (see Section 4.2) that are equipped with Monte Carlo (MC) dropout (Gal & Ghahramani, 2016a), an implicit ensembling technique with a Bayesian motivation that performs training and inference using randomly drawn sub-networks instead of the full deterministic

network. Being motivated as corresponding to Gaussian processes in the limit of infinite layer widths, we shed light on the validity and borders of this equivalence: introducing correlations between the weights and thus the (pre-)activations of the networks (by model training or by correlated initialization), commonly assumed independence assumptions are broken that in turn pose the basis for Gaussianity in the infinite capacity limit (via the central limit theorem). Specifically, we construct randomly initialized networks with global parameter correlations that induce two-sided exponential (instead of Gaussian) output distributions, even for very wide layers. This non-Gaussian dropout modeling regime was—to the best of our knowledge—not systematically studied before.

In the second part of the thesis (Chapter 5), we pick up on dropout-based uncertainty estimation that is promising, as uncertainty information is encoded in the entire structure of the network (which is in contrast to parametric modeling approaches). This characteristic renders stable generalization of dropout-based uncertainty estimates to previously unseen inputs more likely—an important property, given high-dimensional input and output spaces and thus sparse training data samples for many open-world learning systems. Motivated by the previous result that the distributional properties of dropout-induced outputs can be influenced (see Section 4.2), we seek to manipulate the widths of these distributions. This appears worthwhile because the prevailing dropout-based uncertainty technique, MC dropout, does *not* tune the output widths of its implicit sub-network ensemble and instead contracts all sub-networks in all training data points, as it primarily targets the modeling of epistemic uncertainty. In contrast, we optimize the sub-network distribution to match the (implicitly given) output distribution for each training input by minimizing the Wasserstein distance. The resulting uncertainty estimation technique, *Wasserstein dropout* (W-dropout), yields strong uncertainty quality without deteriorating regression performance on a broad set of standard benchmarks. It outperforms not only MC dropout on these datasets but also various state-of-the-art uncertainty techniques. These approaches all rely, to a certain extent, on parametric uncertainty estimates, which is in contrast to W-dropout, which is fully non-parametric.

This property may explain the consistently high stability of W-dropout on a variety of regression tasks, even under continuous and discrete data shifts.[1] As we aim for uncertainty estimates that foster the reliability of real-world ML systems, we finally adapt W-dropout to object detection (OD) architectures, particularly to SqueezeDet (Wu et al., 2017), and find the resulting W-SqueezeDet to outperform MC-SqueezeDet, i.e., an MC-dropout-enhanced version of SqueezeDet. For the RetinaNet architecture, we find these results confirmed. Thus, also for OD, W-dropout is characterized by good uncertainty quality not only when considering averaged scores for in-data test sets but moreover under data shifts—a crucial property for a self-assessment mechanism that is supposed to detect, among others, model insufficiencies. A Gaussian-likelihood one-sample (GL-OS) variant of W-dropout performs similarly well. It reduces, compared to standard W-dropout,

[1] Continuous data shifts (also called drifts) imply small changes to in-data input scenarios, such as newly emerging details (in the case of traffic scenes, e.g., pedestrians on electric scooters), whereas discrete data shifts refer to structurally new inputs (e.g., rural scenarios for a system optimized on urban scenes).

the required amount of training compute, however, at the price of output distributions with a bimodality artifact. As a side result, we propose two uncertainty measures to better quantify uncertainty stability: first, a measure of the depth of distribution tails that enables us to evaluate the likeliness and severity of underestimated uncertainties, i.e., those for which the actual model error is significantly higher than the predicted uncertainty. Second, a measure that considers the entire uncertainty distribution (just like standard calibration scores) that is, however, not calculated in quantile space but in (normalized) residual space and is thus more sensitive to strongly deviating uncertainty estimates. Regarding these two measures, W-dropout outperforms various state-of-the-art uncertainty modeling techniques.

In the third and last part of the work (Chapter 6), we shift the focus from methodological aspects of predicted uncertainties to ML use cases in which uncertainty estimation is employed to increase reliability and thus trustworthiness. Aiming at uncertainty estimators that are tailored to the respective application demands, we propose a conceptual framework for their requirement-based development and testing. Factors that impact uncertainty modeling include regulatory contexts and technical specifications. They determine, for instance, the operational design domain of an uncertainty mechanism, i.e., the regions of the input space where the estimator is supposed to function, and the downstream tasks the estimator is supposed to solve. Technical specifications may introduce additional constraints on the estimator's depth of integration into the ML model, the uncertainty information flow through a surrounding ML model chain, or the granularity of uncertainty estimates. We structure and shape these initial demands by introducing requirement categories that cover aspects of uncertainty quality (e.g., being globally calibrated), conceptual properties (e.g., being argumentatively substantiated), technical characteristics (e.g., coming along with minimal trade-offs), and application-specific requirements (e.g., spatial coherence between pixel-level uncertainty estimates in the case of a model that processes images). Heading for uncertainty acceptance criteria, the quantitative requirements are further formalized as 3-tuples of a semantic data specification, a measure of (uncertainty) quality, and a respective threshold value. Given these desirable uncertainty properties, we guide the selection of an uncertainty mechanism by analyzing how well these desiderata are addressed by widely used uncertainty modeling techniques (and by W-dropout). To determine whether a chosen uncertainty estimator fulfills the respective uncertainty acceptance criteria, a hierarchical test strategy is proposed that is based on iteratively refined (semantic and non-semantic) test data selections. Such novel approaches to test learned uncertainty estimators are required, as established testing strategies for safety-critical applications can hardly be employed for learned systems. These established approaches were often developed for manually assembled systems and (typically) rely on a semantic understanding of a system's (sub-)components that is not given for intrinsically non-interpretable ML systems. Our uncertainty test strategy comprises four hierarchy levels: first, we suggest tests of elementary technical properties based on a coarse data selection, which are followed by tests of global uncertainty properties that build on conceptually broad datasets that are supposed to cover large

parts of the respective operational design domains. The third test category contains subset and pointwise uncertainty tests focusing on specific, e.g., application-critical, regions of input space and quantile ranges of uncertainty estimate and model residual distributions. Complementary uncertainty tests, finally, are an open residual set for "cross-analyses" of uncertainty properties and, moreover, for novel testing methodologies. To apply this test strategy, the depth of testing (within this hierarchy) and its focus points need to be set for each uncertainty acceptance criterion. Given the resulting instantiation of the test strategy, concrete test cases can be derived. Based on the (binary) outcomes of these test cases and a subsequent test result "aggregation" strategy, it is determined whether the use case-specific uncertainty acceptance criteria are (overall) fulfilled by the selected uncertainty estimator.

Discussion and outlook Picking up on the metaphor of a "safety net", the question of its resilience arises, i.e., how reliable uncertainty estimators and downstream safety mechanisms are. This holds, in particular, in the context of modern open-world ML applications that are often based on highly nonlinear (and mathematically hard-to-tame) neural networks. These models also typically operate on high-dimensional data spaces that bring about "unknown unknowns", i.e., entirely novel (high- and low-level) concepts that an ML application needs to handle. Such complex inference scenarios may cause unforeseen model failures or error modes and challenge uncertainty modeling techniques. While approaches like MC dropout come with a theoretical (often Bayesian) motivation that promises, at least to a certain extent, safeguarding against unexpected behavior, we raise awareness of the (inevitably existing) limitations of such theoretical justifications. We analyze (the criticality of) their assumptions, in particular, by explicitly constructing a non-Gaussian "counter"-example of a two-sided exponential output distribution (see Section 4.2). A more precise understanding of when a method's theoretical foundation is stable and when it erodes may increase the confidence in the ML system in the former type of scenario while raising awareness in the latter type of situation. Such information on (evidence for) the validity of underlying (theoretical) assumptions may moreover be used to extend popular summaries of model properties and intended application contexts, such as "model cards" (Mitchell et al., 2019) or "fact sheets" (Arnold et al., 2019).

A central assumption for reliable uncertainty quantification is the use of estimators that are expressive enough to capture the given probabilistic data relations (see, e.g., the DenseHMM in Section 4.1). This is particularly important, as (probabilistic) ML systems in critical applications are supposed to function reliably in *individual* situations, which renders coarse, e.g., dataset-averaged, uncertainty estimates unsuitable. The idea of capturing distributional properties for each individual datapoint motivated the derivation of Wasserstein dropout (see Chapter 5).

While Wasserstein dropout and its one-sample "efficiency" variant focus on the optimization of second moments, they are only two of potentially many ways to actively influence distributional properties of sub-network ensembles. The modeling of second moments is, however, of particular importance, as variances and standard deviations are widely

used as coarse summaries of distributional uncertainty. It therefore seems rewarding to further improve the theoretical basis of W-dropout, specifically, to better understand its empirically observed ability to model epistemic uncertainty. Moreover, one may analyze whether the Bayesian derivation of MC dropout (see Appendix of Gal & Ghahramani (2016a)) can be adapted for heteroscedastic modeling or even for non-Gaussian settings, for instance, by flexibly adjusting the *type* of output distribution to the training data (see, e.g., Meudt et al. (2015); Sick et al. (2020)).

Such methodological developments should be guided by a clear understanding of how to evaluate the resulting uncertainty estimates. To advance the assessment of uncertainties, it seems worthwhile to, on the one hand, better integrate safety aspects into uncertainty scores and, on the other hand, to better comprehend the relationships between the various already existing uncertainty measures. From a safety perspective, for instance, even seemingly minor deviations between often high-dimensional model distributions and data distributions may be critical. This motivates the further development of sensitive (dis-)similarity measures between (samples from) such distributions (like the maximum mean discrepancy (Gretton et al., 2012)). As "manual translations" of complex safety requirements into mathematical quantities are generally challenging, it seems rewarding to moreover consider learning-based approaches to assess probabilistic models. In this respect, one may take inspiration from inverse reinforcement learning (see, e.g., Arora & Doshi (2021)) where objectives and rewards are learned by observing the behavior of an agent (e.g., of a human driver in the case of autonomous driving). However, even when relying on established uncertainty metrics, the dependencies and discrepancies between them (see, e.g., Fig. B.15 (p. 171) in Appendix B) are not yet fully understood, particularly in the field of object detection, where probabilistic models gained interest only recently. Once better understood, these dependencies should be reflected in practitioners' tools for uncertainty assessment that are particularly helpful when "digestible", carefully composed summaries are presented instead of a multitude of unrelated uncertainty scores. The development and use of metrics aside, recent activities in the ML community to strengthen (uncertainty and model) assessments are promising. Positive examples include the publication of datasets with real-world concept shifts (see Koh et al. (2021) and Malinin et al. (2021)) and dedicated conference workshops that focus attention on the topic (see, e.g., the recent ICLR 2022 workshop on ML evaluation standards).

Taking a step back and recalling the general absence of uncertainty ground truth information (see Chapter 1), it seems desirable to improve the data "fundament" of uncertainty quantification, as this may strengthen the development of uncertainty mechanisms and evaluation schemes. Practically feasible ways to generate such extended label information are, for instance, by means of artificial environments or—simple but costly—by labeling real-world datasets multiple times (as was done for the LIDC-IDRI lung image dataset (see Chapter 1) and, recently, for the ImageNet test set (see ImageNet ReaL-H in Tran et al. (2022))). A more scalable, however, approximate approach of varying quality is to learn uncertainty estimates for data labels (see, e.g., Northcutt et al. (2021)).

Moreover, uncertainty information can be extracted from unlabeled data. Specifically, it seems promising to equip self-supervised representation learning approaches with corresponding capabilities (see, e.g., Park et al. (2022)), since a variety of downstream tasks could benefit from learned *probabilistic* embeddings. This holds in particular because latent (deterministic) representations form the backbones of the enormously large neural networks that emerged in recent years and that handle an unprecedented variety of tasks and (in some cases even) data types (Brown et al., 2020; Rae et al., 2021; Ramesh et al., 2022; Alayrac et al., 2022; Reed et al., 2022). Given their broad potential for practical applications, observed overconfident behavior in this new model class (see, e.g., Tab. 8 in Rae et al. (2021)) may have particularly harmful consequences. Self-assessments through reliable uncertainty estimates therefore remain vital to bridge the gap between early and, in part, impressive showcases of these models and actually deployed corresponding learning systems that prove useful and safe in the real world.

Appendix

A. Impact of Model Capacity on Uncertainty

This appendix supplements our analyses on model capacity and uncertainty in Chapter 4. Mirroring the two-part structure of said chapter, we first outline technical details on dense hidden Markov models in Section A.1. Further information on the second part of Chapter 4, Monte Carlo (MC) dropout in wide neural networks, can be found in Section A.2.

A.1. Capacity control of HMMs via representations

Extending Section 4.1 on dense hidden Markov models (DenseHMMs), we first provide the full Lagrangians of standard HMMs and DenseHMMs in Subsection A.1.1. Next, we present the numerical results of nonlinear A-matrix factorizations in Subsection A.1.2. Finally, implementation details and technical aspects of data preprocessing are stated in Subsection A.1.3.

A.1.1. Full Lagrangians of standard HMM and DenseHMM

The full Lagrangian of the standard HMM model in the M-step reads

$$\begin{aligned}\bar{\mathcal{L}} &= \bar{\mathcal{L}}_1 + \bar{\mathcal{L}}_2 + \bar{\mathcal{L}}_3 \\ &= \sum_{i,j\in[n]} \sum_{t=2}^{T} \gamma_t(s_i, s_j) \log a_{ij} + \sum_{i\in[n]} \varphi_i \Big(1 - \sum_{j\in[n]} a_{ij}\Big) \\ &\quad + \sum_{i\in[n]} \sum_{t=1}^{T} \gamma_t(s_i) \log b_{i,j_{o_t}} + \sum_{i\in[n]} \varepsilon_i \Big(1 - \sum_{j\in[m]} b_{ij}\Big) \\ &\quad + \sum_{i\in[n]} \gamma_1(s_i) \log \pi_i + \bar{\varphi} \Big(1 - \sum_{i\in[n]} \pi_i\Big) \,,\end{aligned} \tag{A.1}$$

where j_{o_t} describes the index of the observation observed at time t and $\bar{\varphi}, \varepsilon_i$ are Lagrange multipliers. Applying the transformations $\mathbf{A} = \mathbf{A}(\mathbf{U}, \mathbf{Z})$, $\mathbf{B} = \mathbf{B}(\mathbf{V}, \mathbf{W})$ and $\pi = \pi(\mathbf{U}, \mathbf{z}_{\text{start}})$ yields the full Lagrangian of the DenseHMM:

$$
\begin{aligned}
\bar{\mathcal{L}}^{\text{dense}} &= \bar{\mathcal{L}}_1^{\text{dense}} + \bar{\mathcal{L}}_2^{\text{dense}} + \bar{\mathcal{L}}_3^{\text{dense}} \\
&= \sum_{i,j\in[n]} \sum_{t=2}^{T} \gamma_t(s_i, s_j)\, \boldsymbol{u}_j \cdot \boldsymbol{z}_i - \sum_{i,j\in[n]} \sum_{t=2}^{T} \gamma_t(s_i, s_j) \log \sum_{k\in[n]} \exp(\boldsymbol{u}_k \cdot \boldsymbol{z}_i) \\
&\quad + \sum_{i\in[n]} \sum_{t=1}^{T} \gamma_t(s_i)\, \boldsymbol{v}_{j_{o_t}} \cdot \boldsymbol{w}_i - \sum_{i\in[n]} \sum_{t=1}^{T} \gamma_t(s_i) \log \sum_{j\in[m]} \exp(\boldsymbol{v}_j \cdot \boldsymbol{w}_i) \\
&\quad + \sum_{i\in[n]} \gamma_1(s_i)\, \boldsymbol{u}_i \cdot \boldsymbol{z}_{\text{start}} - \sum_{i\in[n]} \gamma_1(s_i) \log \sum_{j\in[n]} \exp(\boldsymbol{u}_j \cdot \boldsymbol{z}_{\text{start}}) \ .
\end{aligned}
\tag{A.2}
$$

A.1.2. Nonlinear A-matrix factorization

All matrix sizes n and representation lengths l that contribute to the visualized l/n ratios in Fig. 4.3 are shown in Tab. A.1.

A.1.3. Implementation details and data preprocessing

Implementation details The backbone of our implementation is the library `hmmlearn`[1] that provides functions to optimize and score HMMs. The optimization schemes for the DenseHMM models $\mathcal{H}_{\text{dense}}^{\text{EM}}$ and $\mathcal{H}_{\text{dense}}^{\text{direct}}$ are implemented in tensorflow (Abadi et al., 2016). Both models use `tf.train.AdamOptimizer` with a fixed learning rate for optimization. At this point we note that experiments done with other optimizers such that `tf.train.GradientDescentOptimizer` lead to similar results in the evaluation. The representations are initialized using a standard isotropic Gaussian distribution. NLL values are normalized by the number of test sequences and by the maximum test sequence length.

Hardware used All experiments are conducted on a `Intel(R) Xeon(R) Silver 4116 CPU` with `2.10GHz` and a `NVidia Tesla V100`.

Protein dataset preprocessing The first 1,024 sequences of the RCSB PDB dataset have 22 unique symbols. We cut each sequence after a length of 512. Note that less than 4.9% of the 1,024 sequences exceed that length. Additionally, we collect the symbols of lowest frequency that together make up less than 0.2% of all symbols in the sequences and map them onto one residual symbol. This reduces the number of unique symbols in the sequences from 22 to 19.

Part-of-speech sequences preprocessing We take 1,000 sequences from the Medpost dataset (from `tag_mb.ioc`) and cut them after a length of 40, which affects less than 15% of all sequences. We also collect the tags of lowest frequency that together make up less

[1] `https://github.com/hmmlearn/hmmlearn`.

Tab. A.1.: Approximation errors (median with 25/75 percentile) of normAbsLin-based and softmax-based matrix factorizations for different matrix sizes n and representation lengths l.

n	l	median (25/75 percentile) of loss$(\tilde{\mathbf{A}}, \mathbf{A_{gt}})$	
		$\tilde{\mathbf{A}} = \text{normAbsLin}(\mathbf{UZ})$	$\tilde{\mathbf{A}} = \text{softmax}(\mathbf{UZ})$
3	1	0.678 (0.652/0.696)	**0.048** (0.004/0.110)
3	2	0.162 (0.002/0.270)	**0.001** (0.000/0.001)
3	3	**0.001** (0.001/0.001)	**0.001** (0.000/0.001)
3	5	**0.001** (0.001/0.001)	**0.001** (0.001/0.001)
5	1	0.769 (0.745/0.827)	**0.453** (0.321/0.505)
5	3	0.346 (0.093/0.396)	**0.001** (0.001/0.003)
5	5	**0.001** (0.001/0.002)	**0.001** (0.001/0.001)
5	10	**0.001** (0.001/0.002)	0.002 (0.001/0.003)
10	1	0.862 (0.851/0.868)	**0.616** (0.581/0.645)
10	5	0.310 (0.235/0.345)	**0.012** (0.005/0.028)
10	10	**0.002** (0.002/0.002)	0.003 (0.002/0.005)
10	15	**0.002** (0.002/0.002)	0.003 (0.003/0.043)

than 1% of all tags in the sequences and map them onto one residual tag. This reduces the number of tag items from 60 to 42.

Calculation of $\mathbf{\Omega}^{\text{gt}}$ and $\mathbf{\Omega}^{\text{model}}$ The co-occurrence matrices $\mathbf{\Omega}^{\text{model}}$ and $\mathbf{\Omega}^{\text{gt}}$ used to calculate the co-occurrence MADs in Subsection 4.1.4 are estimated by counting subsequent pairs of observation symbols $(o_i(t), o_j(t+1)) \in O^2$. For real-world data, $\mathbf{\Omega}^{\text{gt}}$ is estimated based on the test data ground truth sequences. Equally long sequences sampled from the trained model are used to estimate $\mathbf{\Omega}^{\text{model}}$. In the case of synthetic data, $\mathbf{\Omega}^{\text{gt}}$ is calculated analytically (Eq. 4.11) instead.

A.2. Monte Carlo dropout in wide neural networks

This section contains additional information on our analyses of wide neural networks (see Section 4.2). In particular, we provide implementation details and further results of our numerical experiments on correlations in (un-)trained neural networks in Subsection A.2.1. Detailed analytical calculations for strongly correlated systems can be found in Subsection A.2.2.

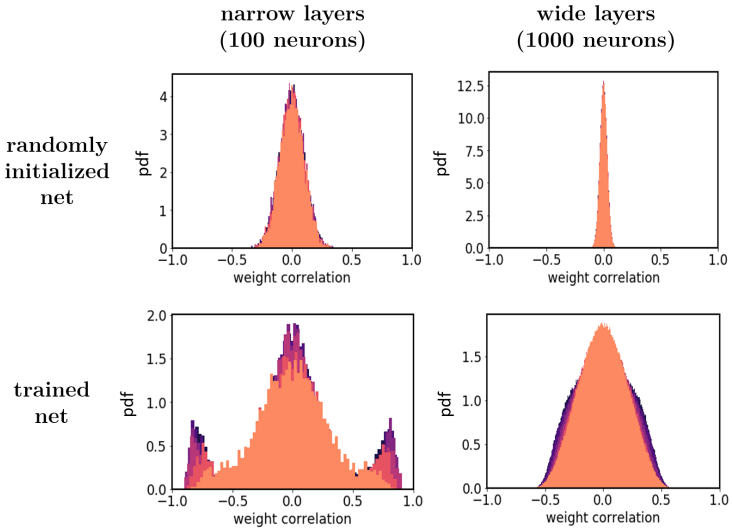


Fig. A.1.: Weight correlations of randomly initialized (top row) and trained (bottom row) networks that are either narrow ($h = 100$, left column) or wide ($h = 1{,}000$, right column). The different hidden layers are color-coded: from first (melon) to last (purple) hidden layer. The Pearson correlation coefficient is used.

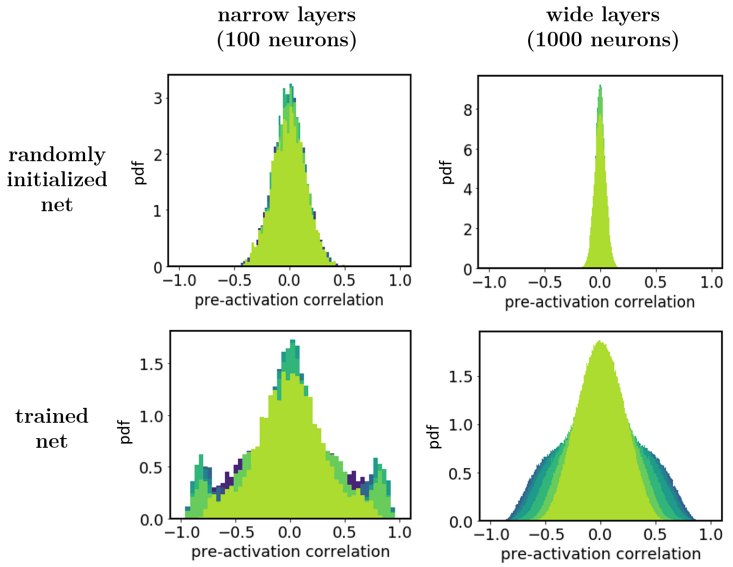


Fig. A.2.: Pre-activation correlations of randomly initialized (top row) and trained (bottom row) networks that are either narrow ($h = 100$, left column) or wide ($h = 1{,}000$, right column). The different hidden layers are color-coded: from first (green) to last (dark blue) hidden layer. The Pearson correlation coefficient is used.

A.2.1. Empirical observations

Further implementation details All biases of the networks are set to zero. Training is done on shuffled mini-batches of size 100 using the standard train-test data split without any further preprocessing. All experiments were run in pytorch (Paszke et al., 2019) using a `Intel(R) Xeon(R) Gold 6126` CPU and a `NVidia GeForce GTX 1080 Ti` GPU.

Correlations in random networks We analyze the weight correlations and pre-activation correlations of the untrained $\mathcal{H}_{\text{narrow}}$ and $\mathcal{H}_{\text{wide}}$. The (row-wise) Pearson correlations of the weight matrices (Fig. A.1, top row) are centered around zero with estimation errors that are determined by the width of the respective network. The column-wise weight correlations look the same. To calculate pre-activation correlations, we run 10,000 forward passes with dropout for a fixed test image. Fig. A.2 (top row) shows that these correlations are largely similar to the weight correlations—as can be expected.

Correlations in trained networks For $\mathcal{H}_{\text{narrow}}$, we find similar weight correlations for all hidden layers with correlations coefficients that range from -1 to 1 (Fig. A.1, bottom left). While the distributions for $\mathcal{H}_{\text{wide}}$ are even more homogeneous across layers, they span only from approximately -0.5 to 0.5 (Fig. A.1, bottom right). More importantly, these distributions resemble the central part of the $\mathcal{H}_{\text{narrow}}$ distributions and do by no means collapse to zero, i.e., although network width varies by one order of magnitude, the global dependence structures are similar in both networks. Next, we study the dependencies between pre-activations that are (mainly) induced by the weight dependencies. Fig. A.2 shows roughly the same dependence pattern as Fig. A.1, however, for $\mathcal{H}_{\text{wide}}$ the pre-activation correlation distribution gets broader with layer depth (Fig. A.2, bottom right). Intuitively, we can make sense of this observation: the network iteratively withdraws input information to finally only keep what is useful for classification, and less information distributed over a fixed number of neurons means stronger correlations. Moreover, not only pre-activation correlations increase with layer depth but also their variances and thus covariances.

Further observations for trained networks We find that sigmoid activations suppress non-normal distributions, which is in contrast to tanh, ReLU, and linear activation functions. Further experiments with a custom nonlinearity that is a proxy to sigmoid suggest that the combination of being constrained and mapping onto only one half-axis might be a critical condition for a Gaussian inducing nonlinearity.

A.2.2. Modeling of strongly correlated systems

Here, we provide additional information accompanying the calculations in Subsection 4.2.3. For instance, the PDF in Eq. 4.19 can be calculated using a filter integral,

$$\begin{aligned}\mathrm{PDF}_{\tilde{f}}(\xi) &\propto \int \mathrm{d}x\,\mathrm{d}y\,\delta(\xi - xy)\,\mathrm{e}^{-x^2/2-y^2/2} \\ &\propto \int \mathrm{d}x\,\mathrm{d}y\,\mathrm{d}k\,\mathrm{e}^{-\mathrm{i}k(\xi-xy)}\,\mathrm{e}^{-x^2/2-y^2/2} \\ &= \int \mathrm{d}k\,\mathrm{e}^{-\mathrm{i}k\xi} \int \mathrm{d}x\,\mathrm{d}y\,\exp\left(-\frac{1}{2}\boldsymbol{b}^T\boldsymbol{A}\,\boldsymbol{b}\right)\end{aligned} \tag{A.3}$$

with the shorthand notations

$$\boldsymbol{b} = \begin{pmatrix} x \\ y \end{pmatrix} \quad \text{and} \quad \boldsymbol{A} = \begin{pmatrix} 1 & -\mathrm{i}k \\ -\mathrm{i}k & 1 \end{pmatrix} \ . \tag{A.4}$$

Evaluating the inner integrals thus leads to

$$\mathrm{PDF}_{\tilde{f}}(\xi) \propto \int_{-\infty}^{+\infty} \mathrm{d}k\,\frac{\mathrm{e}^{-\mathrm{i}k\xi}}{\sqrt{1+k^2}} = 2K_0(|\xi|) \ . \tag{A.5}$$

The differing prefactor in Eq. 4.19 follows from the normalization condition for a PDF. For convenience, we omitted any prefactors here.

A direct extension of this calculation to non-zero mean is challenging. Instead we provide a heuristic explanation where we include this aspect via

$$X = \mu_X + \sigma_X \epsilon_X \quad \text{with } \epsilon_X \sim \mathcal{N}(0,1) \quad (Y \text{ analogous}) \ , \tag{A.6}$$

leading to

$$XY = \underbrace{\mu_X\mu_Y}_{\text{const.}} + \underbrace{\mu_X\sigma_Y\,\epsilon_Y + \mu_Y\sigma_X\,\epsilon_X}_{\text{“Gaussian”}} + \underbrace{\sigma_X\sigma_Y\,\epsilon_X\epsilon_Y}_{\text{“tail”}} \ . \tag{A.7}$$

Neglecting the first term as constant, the two middle summands follow a Gaussian behavior while the last one contains a product giving rise to exponential tails. While this heuristic breakdown ignores the correlations of the twice occurring ϵ's it qualitatively captures the behavior of XY. Indeed, we find a stronger emphasis toward exponential tails for $\sigma > \mu$ and for Gaussian behavior the other way around. For illustration, Fig. A.3 shows the empirical distribution of Z for $\mu_X = \mu_Y = 0$ (left) and $\mu_X = \mu_Y = 10$ (right). As a guide we added a numerically obtained PDF as well as an approximation following the logic of our explanation for Eq. A.7. In this second case we used the addition of two independent random variables $\tilde{Z} = \tilde{X} + \tilde{Y}$, where $\tilde{X} \sim N\left(\mu_X\mu_Y, \sqrt{(\sigma_X\mu_Y)^2 + (\sigma_Y\mu_X)^2}\right)$ is a Gaussian which models the first three terms in Eq. A.7. For $\tilde{Y}$ we use an exponential distribution,

$$\mathrm{PDF}_{\tilde{Y}}(\xi) = \frac{1}{2\sigma_X\sigma_Y}\exp\left(-\frac{|\xi|}{\sigma_X\sigma_Y}\right) \ , \tag{A.8}$$

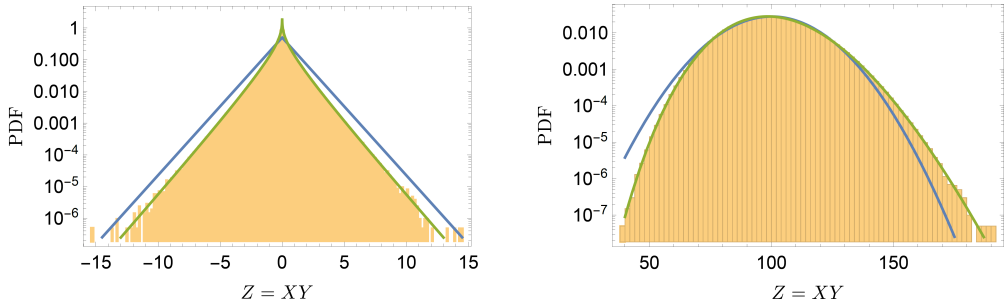


Fig. A.3.: Logarithmic visualization of the PDF for the product of two Gaussian random variables X, Y. Shown in blue and green are an approximation to the PDF (see the text in Section A.2.2) and an exact numerical result, respectively. Left side shows $\mu_X = \mu_Y = 0$ and right $\mu_X = \mu_Y = 10$, in both panels $\sigma_X = \sigma_Y = 1$.

designed to capture the tail properties of the Bessel function. Here, we neglected the additional $|\xi|^{-1/2}$ dependence of the asymptotic, Eq. 4.24, to keep the resulting integrals of a Gaussian type. This allows us to give an explicit expression for $\tilde{Z}$,

$$\begin{aligned}\mathrm{PDF}_{\tilde{Z}}(\xi) = \frac{\mathrm{e}^{\frac{\sigma_1^2}{2\sigma_2^2}}}{4\sigma_2}\Bigg(&\mathrm{e}^{\frac{\mu-\xi}{\sigma_2}}\,\mathrm{Erfc}\left(\frac{\sigma_1^2+(\mu-\xi)\sigma_2}{\sqrt{2}\sigma_1\sigma_2}\right)\\ &+\mathrm{e}^{\frac{\xi-\mu}{\sigma_2}}\,\mathrm{Erfc}\left(\frac{\sigma_1^2+(\xi-\mu)\sigma_2}{\sqrt{2}\sigma_1\sigma_2}\right)\Bigg)\ .\end{aligned} \tag{A.9}$$

Therein we used the short hand notations $\mu = \mu_x\,\mu_Y$, $\sigma_1 = \sqrt{(\sigma_X\mu_Y)^2 + (\sigma_Y\mu_X)^2)}$ and $\sigma_2 = \sigma_X\,\sigma_Y$. Also,

$$\mathrm{Erfc}(\zeta) = 1 - \mathrm{Erf}(\zeta) = 1 - \frac{2}{\sqrt{\pi}}\int_0^\zeta \mathrm{d}t\,\mathrm{e}^{-t^2} \tag{A.10}$$

denotes the complementary error function. As can be seen in Fig. A.3 this approximation roughly captures the exponential tails. Furthermore, we find that for $\mu > \sigma$ the tail behavior is suppressed and the distribution becomes asymmetric, a property we similarly observe in the real data, see Fig. 4.7.

The extension of the $Z = XY$ toy model to a sum of random variables $Z = \sum_{i=1}^{h} X_i Y_i$ can be treated, at least theoretically, in the same way as the original problem:

$$\begin{aligned}\mathrm{PDF}_Z(\xi) &\propto \int \mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{y}\,\delta(\xi - \boldsymbol{x}^T\boldsymbol{y})\,\mathrm{e}^{-\boldsymbol{x}^T\boldsymbol{\Sigma}_X^{-1}\boldsymbol{x}/2 - \boldsymbol{y}^T\boldsymbol{\Sigma}_Y^{-1}\boldsymbol{y}/2}\\ &= \int \mathrm{d}k\,\mathrm{e}^{-\mathrm{i}k\xi}\int \mathrm{d}\boldsymbol{x}\,\mathrm{d}\boldsymbol{y}\,\exp\left(-\frac{1}{2}\boldsymbol{b}^T\boldsymbol{A}\boldsymbol{b}\right)\ ,\end{aligned} \tag{A.11}$$

where in this case $\boldsymbol{b} \in \mathbb{R}^{2h}$ is instead a stacked vector of both $\boldsymbol{x}$ and $\boldsymbol{y}$, compare Eq. A.4. Furthermore,

$$A = \begin{pmatrix} \Sigma_X^{-1} & -\mathrm{i}k\,\mathbb{1} \\ -\mathrm{i}k\,\mathbb{1} & \Sigma_Y^{-1} \end{pmatrix} . \tag{A.12}$$

Evaluating the inner integrals gives rise to

$$\mathrm{PDF}_Z(\xi) \propto \int_{-\infty}^{+\infty} \mathrm{d}k \, \frac{\mathrm{e}^{-\mathrm{i}k\xi}}{\sqrt{\det\left(\Sigma_{\tilde{z}}^{-1}\Sigma_{\tilde{g}}^{-1} + k^2\mathbb{1}\right)}} \ , \tag{A.13}$$

where we used the identity

$$\det \begin{pmatrix} E & B \\ C & D \end{pmatrix} = \det\left(ED - BC\right) \tag{A.14}$$

valid for arbitrary square matrices B, C, D, E as long as $CD - DC = 0$ holds. Assuming that the matrix $\Sigma_{\tilde{z}}^{-1}\Sigma_{\tilde{g}}^{-1}$ is diagonalizable with (positive) eigenvalues σ_i^2 leads to

$$\mathrm{PDF}_Z(\xi) \propto \int_{-\infty}^{+\infty} \mathrm{d}k \, \frac{\mathrm{e}^{-\mathrm{i}k\xi}}{\sqrt{\prod_{i=1}^{h}\left(\sigma_i^2 + k^2\right)}} \ . \tag{A.15}$$

Depending on the eigenvalue spectrum the resulting function can exhibit quite different tail behaviors. To briefly illustrate this, let us make two remarks: If we assume doubly degenerate eigenvalues we can "ignore" the square root and Eq. A.15 instead has poles of *first* order at the positions $k = \pm\mathrm{i}\,\sigma_i$. Based on the residue theorem we then find

$$\mathrm{PDF}_Z(\xi) \propto \sum_{i=1}^{h} \frac{\mathrm{e}^{-\sigma_i|\xi|}}{2\sigma_i \prod_{j\neq i}^{h}(\sigma_j^2 + \sigma_i^2)} \ . \tag{A.16}$$

In a loose sense this might be seen as a discrete variant of a Laplace transform and therefore has a similar variety of outcomes. Those could be largely restricted if one assumes the spectrum of the σ_i to be bounded. In the body of Section 4.2, we omitted this discussion and instead chose an easier example closer to the intended application.

The correlation expressed in Eq. 4.21 can be seen as stemming from a multivariate Gaussian with zero mean and covariance matrix

$$\boldsymbol{\Sigma}_{ij} = \begin{cases} 1 & i = j \\ c & i \neq j \end{cases} \ . \tag{A.17}$$

While this matrix has a very simple structure with only two distinct eigenvalues of $1 + (h-1)c$ and $1 - c$, the latter is $h - 1$ fold degenerate. Therefore, our considerations from Eq. A.16 do not directly apply. Instead, we use this model to build the weight matrices $\boldsymbol{W}_\nu$ for the random network shown in Fig. 4.8. To this end, we draw for each $\boldsymbol{W}_\nu$ two sets of such multivariate Gaussian distributed vectors $\{\boldsymbol{a}_i, \boldsymbol{b}_i \in \mathbb{R}^h\}$. These vectors

form two matrices $\boldsymbol{A} = (a_1, \ldots, a_h) \in \mathbb{R}^{h \times h}$ ($\boldsymbol{B}$ analogous) with correlated rows such that each $\boldsymbol{W}$ is given by $\boldsymbol{W}_\nu = (\boldsymbol{A}_\nu + \boldsymbol{B}_\nu^T)/(2\sqrt{qh})$, where $q = 1 - p$ denotes the keep rate. This way we ensure independent correlations both among rows and columns of the matrices.

B. Modeling Uncertainty Estimates by Means of Wasserstein Dropout

This part accompanies Chapter 5 on Wasserstein dropout and provides further in-depth information. In Section B.1, we present both theoretical and numerical insights into the uncertainties induced by the Gaussian-likelihood one-sample (GL-OS) variant of Wasserstein dropout. We then shift the focus to "standard" Wasserstein dropout, which is our object of study in the remainder of this appendix. Large parts of its empirical evaluation on toy data and standard regression datasets can be found in Section B.2, including details on the datasets, more granular evaluations, and additional toy data experiments. Details on the object detection datasets and supplementary evaluations of SqueezeDet are located in Subsection B.2.5. As W-dropout exhibits the hyperparameters p (drop rate) and L (sample size), we test various values in Section B.3, finding no strong correlation between result and parameter choices. We close with a discussion on the relation between uncertainty measures and their respective sensitivity in Section B.4.

B.1. Detailed analysis of the Gaussian-likelihood one-sample variant of Wasserstein dropout

In Section 5.1, we observed that the Gaussian-likelihood one-sample (GL-OS) variant of Wasserstein dropout may induce bimodal output distributions. A theoretical analysis of this behavior is provided in Subsection B.1.1. The uncertainty estimates of GL-OS Wasserstein dropout are not only encoded in the widths of these (bimodal) output distributions but moreover in the distances between the deterministic and the (mean) probabilistic outputs of the networks. This composition of the uncertainty estimates is empirically studied in Subsection B.1.2. Finally, we visualize different components of the GL-OS objective for trained models to gain insights into the concurrent optimization of uncertainty estimates and regression performance during model training (see Subsection B.1.3).

B.1.1. Analytical properties of the GL-OS Wasserstein dropout loss

In the following, we look closer at the behavior of the GL-OS Wasserstein dropout loss with respect to aleatoric uncertainty. For this, we assume that the residuals (see Eq. 5.9) are given by a Gaussian distribution with, for simplicity, $\mu_{\text{Res.}} = 0$ and $\sigma_{\text{Res.}} = 1$. We want to determine the resulting loss for the second summand in Eq. 5.9, termed L_2, that governs the uncertainty estimation of the model.[1] It depends on the underlying distribution of

[1] In this subsection, the argument x_i is omitted for clarity of exposition.

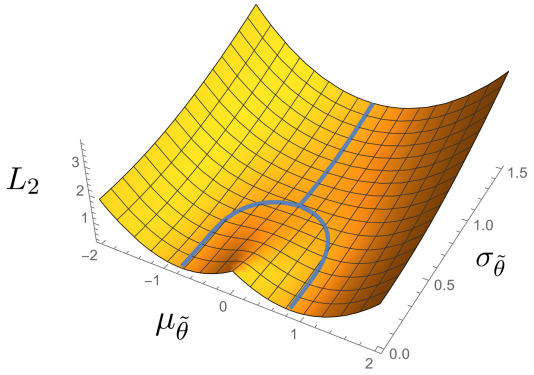


Fig. B.1.: Shown is the value of the second loss component in Eq. 5.9, termed L_2, over $\mu_{\tilde{\boldsymbol{\theta}}}$ and $\sigma_{\tilde{\boldsymbol{\theta}}}$ that describe the implicit dropout ensemble. The blue line shows the position of the minima of L_2 for fixed values of $\sigma_{\tilde{\boldsymbol{\theta}}}$. Clearly visible are the global minima at $\sigma_{\tilde{\boldsymbol{\theta}}} = 0$ and the bifurcation at $\sigma_{\tilde{\boldsymbol{\theta}}} = 2/\pi$.

the effective MC dropout distribution, which we model as $\mathcal{N}(\mu_{\tilde{\boldsymbol{\theta}}}, \sigma_{\tilde{\boldsymbol{\theta}}})$ such that

$$L_2 = \int_{-\infty}^{\infty} \mathrm{d}y_1 \mathrm{d}y_2 \left(|y_1| - |y_2|\right)^2 p_1(y_1)\, p_2(y_2)\,, \tag{B.1}$$

where p_1 and p_2 are the Gaussian distributions discussed above. After some calculation this yields

$$L_2 = -\frac{4}{\pi}\sigma_{\tilde{\boldsymbol{\theta}}} \exp\left(-\frac{1}{2}\frac{\mu_{\tilde{\boldsymbol{\theta}}}^2}{\sigma_{\tilde{\boldsymbol{\theta}}}^2}\right) - \sqrt{\frac{8}{\pi}}\,\mu_{\tilde{\boldsymbol{\theta}}}\ \mathrm{Erf}\left(\frac{\mu_{\tilde{\boldsymbol{\theta}}}}{\sqrt{2}\,\sigma_{\tilde{\boldsymbol{\theta}}}}\right) + \sigma_{\tilde{\boldsymbol{\theta}}}^2 + \mu_{\tilde{\boldsymbol{\theta}}}^2 + 1\,, \tag{B.2}$$

which is visualized in Fig. B.1. The two global minima can be found for $(\mu_{\tilde{\boldsymbol{\theta}}}, \sigma_{\tilde{\boldsymbol{\theta}}}) = (\pm\sqrt{2/\pi}, 0)$. However, as we model a randomized residual y_1 these minima do not reach zero. We find that it is favorable to move $\mu_{\tilde{\boldsymbol{\theta}}}$ away from the network prediction of $\mu_{\text{Res.}} = 0$, the mean of the underlying data distribution. But, this is only the case as long as the inherent uncertainty in the dropout distribution can be brought below $\sigma_{\tilde{\boldsymbol{\theta}}} < 2/\pi$, which is still smaller than the uncertainty of $\sigma_{\text{Res.}} = 1$ assumed within the training data distribution. Otherwise, it is more favorable to have $\mu_{\tilde{\boldsymbol{\theta}}} = \mu_{\text{Res.}} = 0$. In the following sections we investigate the practical implications of this finding. For instance, as detailed in Subsection B.2.2, the highly oscillating or noisy toy model experiments clearly exhibited the type of separation discussed here. Decomposing the uncertainty for the standard 1D regression datasets in Subsection B.1.2, on the other hand, showed mixed behavior with indications for bimodal shifts in $\mu_{\tilde{\boldsymbol{\theta}}}$ as well as improved values of $\sigma_{\tilde{\boldsymbol{\theta}}}$.

We already showed the effect of this bimodality in Fig. 5.1 in Section 5.1, where various sub-networks where sampled. Clearly visible is a stronger variation between the networks compared to MC, but also a concentration around the two possible minima. While Fig. 5.1 provides a good visual estimate of $\sigma_{\tilde{\boldsymbol{\theta}}}$, the total uncertainty σ_{total} would additionally

contain the systematic shift $|f_{\boldsymbol{\theta}} - \mu_{\tilde{\boldsymbol{\theta}}}|$. Given the roughly symmetric distribution of the sub-networks we can expect it to be comparatively small.

B.1.2. Composition of the uncertainty estimate

The uncertainty estimate of the GL-OS Wasserstein dropout loss is comprised of two parts: $\sigma_{\text{total}}(x_i) = \sigma_{\tilde{\boldsymbol{\theta}}}(x_i) + |f_{\boldsymbol{\theta}}(x_i) - \mu_{\tilde{\boldsymbol{\theta}}}(x_i)|$. Fig. B.2 reveals that $\sigma_{\tilde{\boldsymbol{\theta}}}(x_i)$ contributes to more than 80% of $\sigma_{\text{total}}(x_i)$ for the three presented datasets and for all applied data splits. A highly similar behavior can be observed for all other datasets. The analytical consideration in Appendix B.1.1 suggests that for cases without data-inherent uncertainty the GL-OS Wasserstein dropout loss provides no incentive for $|f_{\boldsymbol{\theta}}(x_i) - \mu_{\tilde{\boldsymbol{\theta}}}(x_i)| > 0$. The same holds true in the presence of aleatoric uncertainty as long as $\sigma_{\tilde{\boldsymbol{\theta}}}(x_i)$ is comparably large. For aleatoric uncertainty and small $\sigma_{\tilde{\boldsymbol{\theta}}}(x_i)$ larger $|f_{\boldsymbol{\theta}}(x_i) - \mu_{\tilde{\boldsymbol{\theta}}}(x_i)|$ are favorable. However, as our loss is radial symmetric, all directions are equivalent and initialization and randomness determine the direction of the spread $|f_{\boldsymbol{\theta}}(x_i) - \mu_{\tilde{\boldsymbol{\theta}}}(x_i)|$ for each individual sub-network. This symmetry leads again to a small averaged $|f_{\boldsymbol{\theta}}(x_i) - \mu_{\tilde{\boldsymbol{\theta}}}(x_i)|$. The term $\sigma_{\tilde{\boldsymbol{\theta}}}(x_i)$ on the contrary describes the width of a bimodal set of sub-networks in these cases.

B.1.3. Detailed analysis of the two loss components

A deeper look into the structure of the GL-OS Wasserstein dropout loss is possible if we investigate its behavior component-wise. To clarify the results presented in Fig. B.3, we recall the loss structure as $L = L_1 + L_2 = \sum_{i=1}^{M} \left[a_i^2 + (|b_i| - |a_i|)^2 \right]$ with $a_i = f_{\boldsymbol{\theta}}(x_i) - y_i$ and $b_i = f_{\tilde{\boldsymbol{\theta}}}(x_i) - f_{\boldsymbol{\theta}}(x_i)$. Histograms of the a_i (Fig. B.3, first column) enable a detailed view on network performance. The uncertainty quality of the networks can be judged by studying the L_2 loss term more closely, namely, by visualizing histograms of $|b_i| - |a_i|$ (fourth column). The second and third column zoom into L_2 and show histograms of the b_i and scatter plots of (b_i, a_i), respectively. Only test datasets are visualized and as we applied 90 : 10 train-test splits, this explains the low resolution of some histograms in the first column. All quantities involving b_i require the sampling of sub-networks. We draw 200 sub-networks. This sampling procedure explains the higher plot resolutions in columns two to four.

Qualitatively, we observe that both the a_i's and b_i's are centered around zero, which hints at successful optimization of regression performance and of uncertainty quality. Details on how the optimization is realized on a technical level can be gained from the scatter plots. They show three qualitative shapes: a "cross" (first row), a "line" (second row) and a "blob" (third and fourth row). The "cross" occurs for "toy-hf" and reflects the bimodal sub-network structure we found in Fig. 5.1. For an in-detail discussion of the uni- and bimodality of the GL-OS Wasserstein dropout loss landscape, see Appendix B.1.1. A "line" shape reflects that all sub-networks occupy the same minimum given a bimodal case. Following Appendix B.1.1, a "blob" indicates a unimodal case that might be evoked by large standard deviations $\sigma_{\tilde{\boldsymbol{\theta}}}(x_i)$.

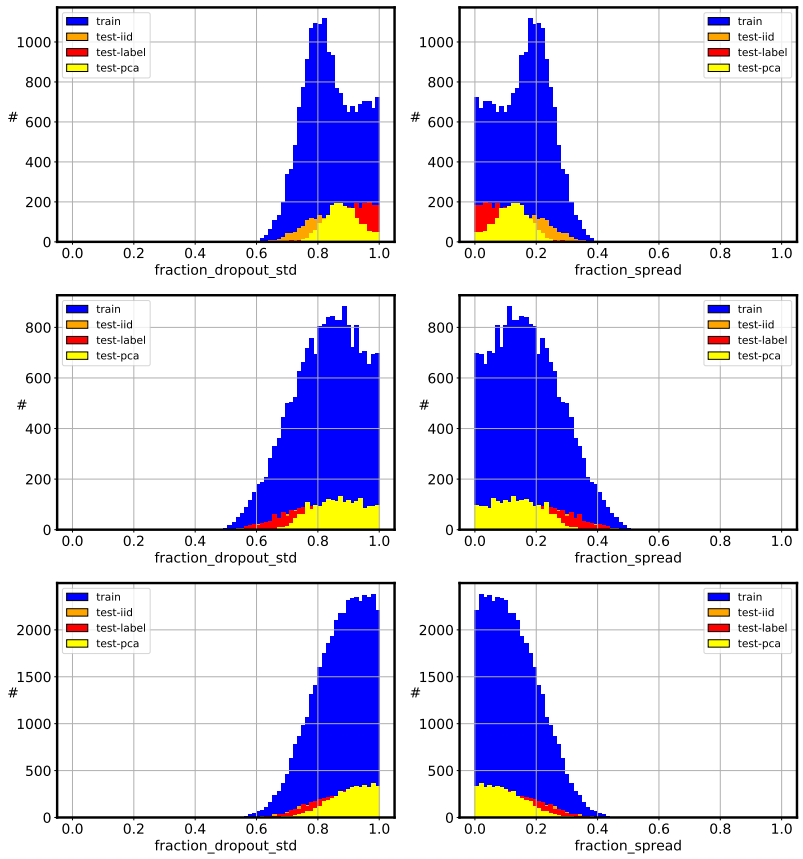


Fig. B.2.: The GL-OS Wasserstein dropout loss induces uncertainties $\sigma_{\text{total}}(x_i) = \sigma_{\tilde{\boldsymbol{\theta}}}(x_i) + |f_{\boldsymbol{\theta}}(x_i) - \mu_{\tilde{\boldsymbol{\theta}}}(x_i)|$. The relative contribution of both components ("fraction_dropout_std", "fraction_spread") is shown for three exemplary datasets (top: "toy-noise", middle: "superconduct", bottom: "protein") and i.i.d. (train: blue, test: orange) as well as non-i.i.d. data splits (test-label: red, test-pca: yellow).

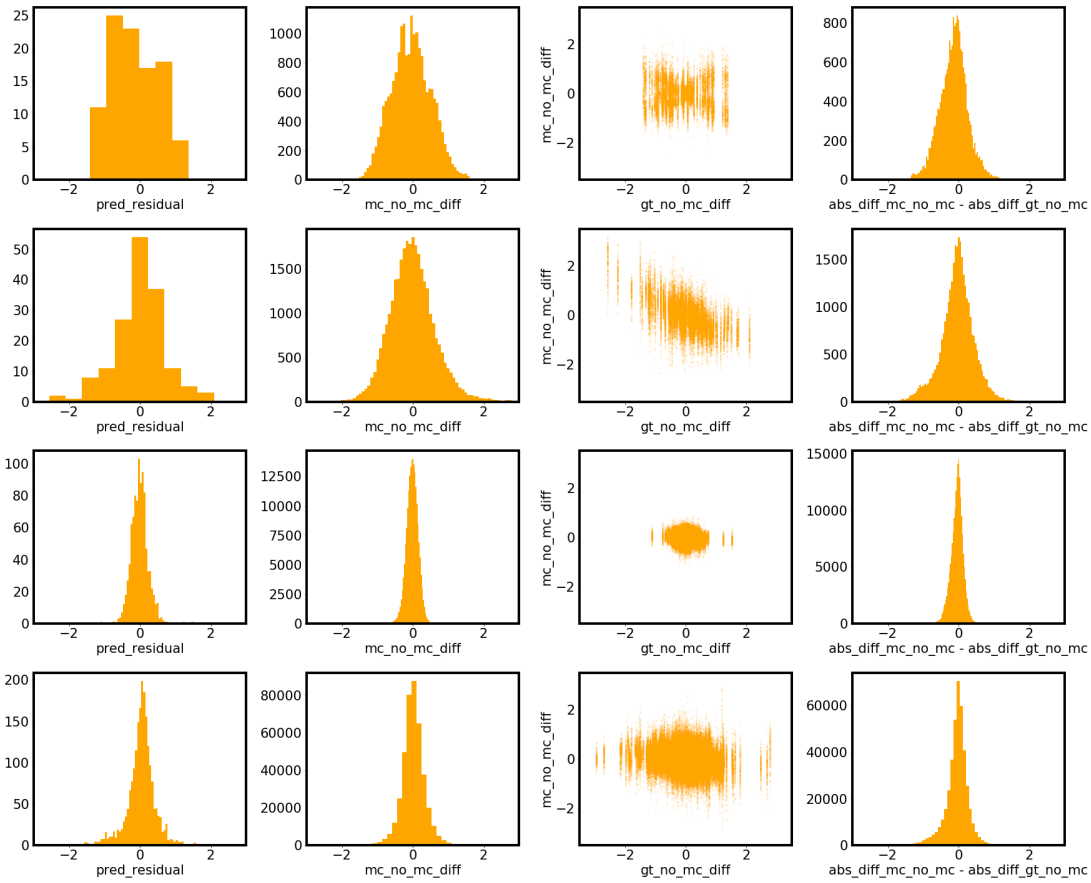


Fig. B.3.: Visualization of the components (columns) of the GL-OS Wasserstein dropout loss for selected test datasets (rows). The prediction residual $f_{\boldsymbol{\theta}}(x_i) - y_i$ (first column), model spread $f_{\tilde{\boldsymbol{\theta}}}(x_i) - f_{\boldsymbol{\theta}}(x_i)$ (second column), a scatter plot of both quantities (third column) and $|f_{\tilde{\boldsymbol{\theta}}}(x_i) - f_{\boldsymbol{\theta}}(x_i)| - |f_{\boldsymbol{\theta}}(x_i) - y_i|$ (fourth column) are shown. The chosen datasets from top to bottom are: "toy-hf", "wine-red", "power" and "california".

B.2. Extension to the empirical study

Complementing the evaluation sketched in Section 5.3, we provide more details on the training setup and benchmark approaches in the following subsection. Further information on the toy dataset experiments can be found in Subsection B.2.2. The same holds for the 1D regression experiments in Subsection B.2.3, which we extend by evaluations on dataset level that were skipped in the main text. A close look at the predicted uncertainties (per method) on these datasets is given via scatter plots in Subsection B.2.4. Details on OD dataset preprocessing and SqueezeDet results are found in the last subsection.

B.2.1. Experimental setup

The experimental setup used for the toy data and 1D regression experiments is presented in two parts: first, technical details of the benchmark approaches we compare with and second, a description of the neural networks and training procedures we employ.

For MC dropout, we choose the regularization coefficient λ by grid search on the set $\lambda \in \{0, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$ and find $\lambda = 10^{-6}$ to provide the best overall results for the 1D regression datasets. A variant of MC dropout that optimizes its layer-specific drop rates during training is Concrete dropout (CON-MC): all its initial drop rates are set to $p_{\text{initial}} = 0.1$. The hyperparameters $wr = l^2/(\tau N)$ and $dr = 2/N$ are determined by the number of training data points N, prior length scale $l = 10^{-3}$ and $\tau(N) \in [10^{-3}, 2]$ that decreases monotonically with N. For PU and PU-EV networks, we ensure positivity constraints using softplus (Glorot et al., 2011) and optimize Gaussian NLL and t-distribution NLL, respectively. The regularization coefficient of PU-EV is set to $\lambda = 10^{-2}$, determined by a grid search considering the parameter range $\lambda \in \{10^{-4}, 10^{-3}, 10^{-2}, 0.1, 0.5\}$. For SWAG, we start to estimate the low-rank Gaussian proxy (rank $r = 20$) for the NN weight distribution after training for $n/2$ epochs, with n being the total number of training epochs.

We categorize the toy and 1D regression datasets as follows: small datasets {“toy-hf”, “yacht”, “diabetes”, “boston”, “energy”, “concrete”, “wine-red”}, large datasets {“toy-noise”, “abalone”, “kin8nm”, “power”, “naval”, “california”, “superconduct”, “protein”} and very large datasets {“year”}. For small datasets, NNs are trained for 1,000 epochs using mini-batches of size 100. All results are 10-fold cross-validated. For large datasets, we train for 150 epochs and apply 5-fold cross validation. We keep this large-dataset setting for the very large “year” dataset but increase mini-batch size to 500.

All experiments are conducted on `Core Intel(R) Xeon(R) Gold 6126` CPUs and `NVidia Tesla V100` GPUs. Conducting the described experiments with cross validation on one CPU takes $20\,h$ for toy data, $130\,h$ for 1D regression datasets and approximately $100\,h$ for object regression on the GPU.

B.2.2. Toy datasets: systematic evaluation and further experiments

The “toy-noise” and “toy-hf” datasets are sampled from $f_{\text{noise}}(x) \sim \mathcal{N}(0, \exp(-0.02\,x^2))$ for $x \in [-15, 15]$ and $f_{\text{hf}}(x) = 0.25\,x^2 - 0.01\,x^3 + 40\,\exp(-(x+1)^2/200)\,\sin(3\,x)$ for

Tab. B.1.: Regression performance and uncertainty quality of networks with different uncertainty mechanisms. All scores are calculated on the test set of "toy-hf" and "toy-noise", respectively.

measure	dataset	swag	de	pu	pu-ev	pu-de
RMSE (↓)	toy-hf	0.696	0.660	0.691	0.691	0.690
NLL (↓)	toy-hf	85.331	52.444	−0.098	1.855	−0.100
ECE (↓)	toy-hf	1.472	1.584	0.548	0.500	0.524
WS (↓)	toy-hf	9.043	7.413	0.233	0.242	0.243
RMSE (↓)	toy-noise	1.006	1.006	1.006	1.006	1.006
NLL (↓)	toy-noise	6934.498	$1.14 \cdot 10^4$	−0.374	1.555	−0.374
ECE (↓)	toy-noise	1.541	1.642	0.062	0.098	0.084
WS (↓)	toy-noise	63.760	83.590	0.028	0.064	0.048

measure	dataset	pu-mc	con-mc	mc	w-drop
RMSE (↓)	toy-hf	0.694	0.701	0.696	0.678
NLL (↓)	toy-hf	−0.083	17.616	13.370	−0.055
ECE (↓)	toy-hf	0.544	1.380	1.352	0.428
WS (↓)	toy-hf	0.233	4.356	3.830	0.222
RMSE (↓)	toy-noise	1.007	0.995	1.006	1.013
NLL (↓)	toy-noise	−0.370	$6.57 \cdot 10^4$	1.723	−0.330
ECE (↓)	toy-noise	0.066	1.730	0.645	0.107
WS (↓)	toy-noise	0.030	$5.03 \cdot 10^4$	0.693	0.054

$x \in [-15, 20]$, respectively. Standard normalization is applied to input and output values. Detailed evaluations of the considered uncertainty methods on these datasets are given in Tab. B.1.

To illustrate the capabilities and limitations of MC dropout regarding the modeling of aleatoric uncertainty, we consider the "toy-noise" dataset again and systematically vary MC's regularization parameter λ (see Fig. B.4, λ decreases from left to right). As MC dropout's uncertainty estimates contain an additive constant term proportional to λ, tuning this parameter allows for modeling the *average* aleatoric uncertainty (the ideal λ in Fig. B.4 is between $\lambda = 10^{-6}$ and $\lambda = 10^{-5}$). Input dependencies of noise (heteroscedasticity) can, however, not be incorporated, i.e., even an optimized λ causes systematic over- and under-estimations of the data uncertainty in many cases. This is in contrast to W-dropout.

Having shown that W-dropout can approximate input-dependent data uncertainty appropriately (see Fig. 5.1), we now analyze its ability to match ground truth uncertainties σ_{true} more systematically. Therefore, we fit a "noisy line" toy dataset that is given by

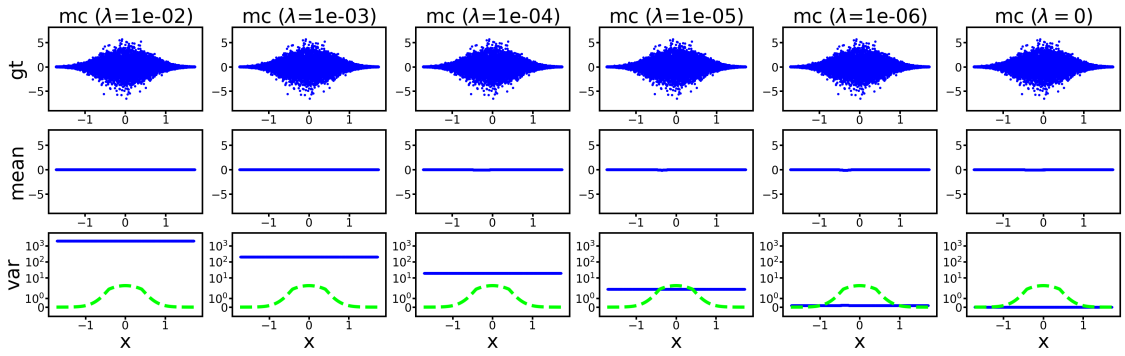


Fig. B.4.: MC dropout and aleatoric uncertainty. The regularization parameter λ of MC dropout allows us to model the average (homoscedastic) noise level of a dataset. As the regularizer is not input-dependent, it does not capture the x-dependency of the noise level, i.e., the heteroscedasticity of the dataset, see third row.

(x_i, y_i) with $x_i \sim \mathcal{U}(-1, 1)$ and $y_i \sim \mathcal{N}(0, \sigma_{\text{true}})$. The ground truth standard deviations take the values $\sigma_{\text{true}} = 0, 0.1, 0.2, 0.5, 1, 2, 5, 10$. Fig. B.5 emphasizes that W-dropout provides accurate uncertainty estimates for both small and large noise levels. Minor x-dependent fluctuations (see whiskers in Fig. B.5) decrease monotonically with σ_{true}.

B.2.3. Standard regression datasets: systematic evaluation

An overview on the 1D regression datasets providing basic statistics and information on preprocessing is given in Tab. B.2. Evaluations of RMSE, NLL, ECE and WS on dataset level can be found in Tab. B.3. Moreover, we extend our evaluation by a deterministic network that we obtain as one of the members of the DE ensemble. For better overview, we reproduce Fig. 5.6 (top row) with this member added, see Fig. B.6. The small performance deterioration of this member compared to the full DE ensemble can be attributed to the averaging over the outcomes of the five ensemble components that suppresses stochastic fluctuations.

B.2.4. Residual-uncertainty scatter plots

Visual inspection of uncertainties can be helpful to understand their qualitative behavior. We scatter model residuals $\mu_i - y_i$ (respective x-axis in Fig. B.8) against model uncertainties σ_i (resp. y-axis in Fig. B.8). For a *hypothetical ideal* uncertainty mechanism, we expect $(y_i - \mu_i) \sim \mathcal{N}(0, \sigma_i)$, i.e., model residuals following the predictive uncertainty distribution. More concretely, 68.3% of all $(y_i - \mu_i)$ would lie within the respective interval $[-\sigma_i, \sigma_i]$ and 99.7% of all $(y_i - \mu_i)$ within $[-3\,\sigma_i, 3\,\sigma_i]$. Fig. B.7 visualizes this hypothetical ideal. It is generated as follows: We draw 3,000 standard deviations $\sigma_i \sim \mathcal{U}(0, 2)$ and sample residuals r_i from the respective normal distributions, $r_i \sim \mathcal{N}(0, \sigma_i)$. The pairs (r_i, σ_i)

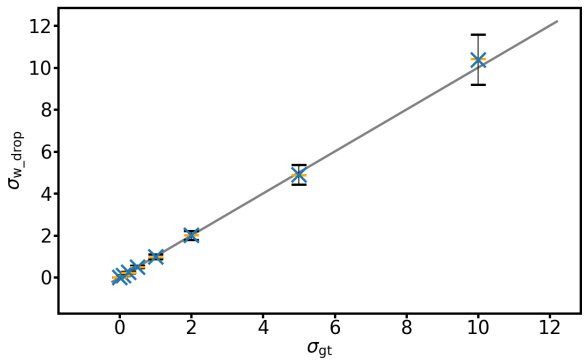


Fig. B.5.: Standard deviation $\sigma_{\text{w-drop}}$ of W-dropout (y-axis) when fitted to a toy dataset with ground truth standard deviation σ_{gt} (x-axis, see the text in Subsection B.2.2 for details). The bisecting line is shown in gray. While $\sigma_{\text{w-drop}}$ exhibits fluctuations (black whiskers at 10% and 90% quantile), it provides on average accurate estimates of the ground truth uncertainty. Both mean value (blue cross) and median value (orange bar) of $\sigma_{\text{w-drop}}$ are close to the bisector.

are visualized. By construction, uncertainty estimates now ideally match residuals in a distributional sense.

Geometrically, the described Gaussian properties imply that 99.7% of all scatter points, e.g., in Fig. B.8, should lie above the blue 3σ lines and 68.3% of them above the yellow 1σ lines. For "toy-noise", "abalone" and "superconduct" (first, third and fourth row in Fig. B.8), PU, PU-DE and W-dropout qualitatively fulfill this requirement while MC, MC-LL and DE tend to underestimate uncertainties. This finding is in accordance with our systematic evaluation. The "naval" dataset (second row in Fig. B.8) poses an exception in this regard as all uncertainty methods lead to comparably convincing uncertainty estimates. The small test RMSEs of all methods on "naval" indicate relatively small aleatoric uncertainties and model residuals. Epistemic uncertainty might thus be a key driving factor and coherently MC, MC-LL and DE perform well.

B.2.5. Object detection: systematic evaluation

We report basic information on the object detection (OD) datasets and their harmonization in the first paragraph of this subsection. Supplementary evaluations of SqueezeDet can be found subsequently in the second paragraph.

Details on OD datasets The six OD datasets we consider are diverse in multiple dimensions as they capture traffic scenes from three continents (Asia, Europe and North America) and cover a broad set of scenarios ranging from cities and metropolitan areas over country roads to highways (see Tab. B.5). They moreover differ in the average number of objects per image (see Tab. 5.4) that reaches its highest values for the simulation-

Tab. B.2.: Details on 1D regression datasets. The ground truth annotations are partially preprocessed to match the 1D regression setup: for the "energy" and the "naval" dataset, we use only the "cooling load" and the "turbine" label, respectively. For the "abalone" dataset, we omit the categorical input feature "sex".

dataset	source	# features	# data points
yacht	UCI	6	308
diabetes	StatLib	7	442
boston	StatLib	13	506
energy	UCI	8	768
concrete	UCI	8	1,030
wine-red	UCI	11	1,599
abalone	UCI	7	4,176
kin8nm	Delve	8	8,192
power	UCI	4	9,568
naval	UCI	16	11,934
california	StatLib	8	20,640
superconduct	UCI	81	21,263
protein	UCI	9	45,730
year	UCI	90	515,345

based SynScapes dataset.[2] Finally, both random and sequence-based train-test splits are considered. This variety is moreover reflected in the numerous object classes the different datasets provide. Their mappings to three main categories ("pedestrian", "cyclist", "vehicle") can be found in Tab. B.6. Rare or irregular classes are removed. For KITTI, we moreover discard "van", "truck" and "person-sitting", following the original SqueezeDet paper. To analyze uncertainty quality on distorted images, blurred and noisy versions of the test datasets are created. Fig. B.10 shows these transformations for two exemplary images from BDD100k (top row) and SynScapes (bottom row), respectively.

Comparison with the deterministic SqueezeDet A comparison of the results of the deterministic SqueezeDet, MC-SqueezeDet and W-SqueezeDet for the KITTI test dataset can be found in Tab. B.4. There, we additionally consider the mean average precision (mAP) score of these networks as well as the closely related measures of precision and recall.[3] In contrast to the regression results on the 1D standard datasets, we observe more pronounced deviations between the different types of SqueezeDet and, in particular,

[2] For A2D2, 2D bounding boxes are inferred from semantic segmentation ground truth.

[3] In object detection, average precision (AP) can be understood as the area under the precision-recall curve that is obtained when sorting all predicted bounding boxes (for a given dataset) by their confidence scores.

Tab. B.3.: Regression performance and uncertainty quality of networks with different uncertainty mechanisms. The scores are calculated on the test sets of 14 standard regression datasets. Please note that the deterministic network does not provide uncertainty information.

measure	dataset	deterministic	swag	de	pu	pu-ev	pu-de	pu-mc	con-mc	mc	w-drop
RMSE (↓)	yacht	0.045	0.055	0.042	0.062	0.051	0.049	0.076	0.103	0.074	0.075
NLL (↓)	yacht	–	−3.010	−3.560	−2.370	−0.438	−3.017	−2.881	−2.059	−2.202	−2.489
ECE (↓)	yacht	–	0.757	0.653	0.863	0.890	1.087	1.093	1.010	1.093	1.113
WS (↓)	yacht	–	0.470	0.379	0.668	1.228	0.544	0.560	0.481	0.536	0.537
RMSE (↓)	diabetes	0.971	0.990	0.920	0.816	0.832	0.790	0.773	0.792	0.843	0.886
NLL (↓)	diabetes	–	61.217	7.629	1712.520	11.505	7.423	2.302	3.427	6.014	2.356
ECE (↓)	diabetes	–	1.456	0.942	1.353	1.370	0.945	0.854	0.975	1.108	0.909
WS (↓)	diabetes	–	7.704	1.717	19.552	19.182	1.936	1.230	1.630	2.229	1.265
RMSE (↓)	boston	0.345	0.352	0.322	0.339	0.349	0.329	0.287	0.317	0.310	0.318
NLL (↓)	boston	–	10.739	6.116	553.188	4.003	3.590	0.050	−0.371	−0.040	−0.287
ECE (↓)	boston	–	1.170	0.882	1.294	1.240	0.876	0.644	0.636	0.688	0.624
WS (↓)	boston	–	2.987	1.593	9.233	5.288	1.490	0.658	0.493	0.674	0.581
RMSE (↓)	energy	0.097	0.077	0.118	0.166	0.106	0.079	0.107	0.080	0.075	
NLL (↓)	energy	–	−1.048	0.628	5.882	1.751	−2.007	−2.056	−1.703	−1.815	−2.047
ECE (↓)	energy	–	0.567	0.602	1.037	0.819	0.502	0.705	0.637	0.819	0.648
WS (↓)	energy	–	0.672	0.723	2.183	1.508	0.374	0.379	0.332	0.424	0.339
RMSE (↓)	concrete	0.258	0.258	0.241	0.263	0.260	0.244	0.237	0.270	0.235	0.248
NLL (↓)	concrete	–	2.726	4.899	34.402	2.496	0.208	−0.978	−0.672	−0.890	−0.868
ECE (↓)	concrete	–	0.672	0.649	0.871	0.763	0.528	0.384	0.395	0.431	0.400
WS (↓)	concrete	–	1.279	1.402	3.217	2.605	0.773	0.230	0.256	0.267	0.253
RMSE (↓)	wine-red	0.981	0.931	0.835	0.824	0.817	0.771	0.783	0.748	0.784	0.807
NLL (↓)	wine-red	–	10.343	1.352	$1.41 \cdot 10^5$	10.201	3.578	11.142	2.485	1.962	0.830
ECE (↓)	wine-red	–	0.946	0.488	0.765	0.816	0.456	0.541	0.677	0.664	0.549
WS (↓)	wine-red	–	2.487	0.717	35.052	55.450	0.813	0.914	1.154	1.014	0.536
RMSE (↓)	abalone	0.673	0.798	0.701	0.653	0.657	0.654	0.636	0.636	0.684	0.718
NLL (↓)	abalone	–	13.289	31.002	$2.05 \cdot 10^8$	2.486	−0.073	−0.111	9.206	1.017	0.256
ECE (↓)	abalone	–	1.181	1.276	0.252	0.250	0.238	0.270	1.082	0.496	0.402
WS (↓)	abalone	–	3.149	4.624	$3.07 \cdot 10^7$	286.448	0.157	0.141	2.630	0.726	0.462
RMSE (↓)	kin8nm	0.269	0.259	0.246	0.272	0.276	0.253	0.269	0.313	0.266	0.261
NLL (↓)	kin8nm	–	−0.393	1.905	−0.142	1.274	−0.866	−0.631	−0.612	−0.678	−0.855
ECE (↓)	kin8nm	–	0.453	0.677	0.462	0.362	0.205	0.561	0.223	0.491	0.151
WS (↓)	kin8nm	–	0.552	1.202	0.629	0.405	0.185	0.334	0.203	0.306	0.080
RMSE (↓)	power	0.224	0.228	0.219	0.225	0.227	0.221	0.228	0.234	0.226	0.226
NLL (↓)	power	–	1.682	12.696	−0.921	0.870	−1.024	−1.007	−0.470	−0.720	−0.788
ECE (↓)	power	–	0.755	1.075	0.154	0.176	0.135	0.172	0.415	0.656	0.264
WS (↓)	power	–	1.306	2.943	0.127	0.193	0.071	0.098	0.520	0.393	0.318
RMSE (↓)	naval	0.087	0.250	0.030	0.163	0.194	0.165	0.169	0.345	0.118	0.100
NLL (↓)	naval	–	−0.198	−2.815	−2.358	0.327	−1.405	−1.464	−0.522	−0.701	−1.479
ECE (↓)	naval	–	1.230	0.898	0.677	0.887	1.235	0.760	0.378	1.233	0.888
WS (↓)	naval	–	0.615	0.487	0.357	0.931	0.606	0.454	0.297	0.632	0.483
RMSE (↓)	california	0.453	0.444	0.430	0.674	0.514	0.475	0.549	0.456	0.436	0.448
NLL (↓)	california	–	0.813	7.305	−0.494	1.266	−0.612	−0.560	1.441	−0.236	−0.213
ECE (↓)	california	–	0.469	0.829	0.248	0.212	0.251	0.314	0.625	0.650	0.251
WS (↓)	california	–	0.745	1.965	0.181	306.628	0.162	0.194	0.989	0.357	0.312
RMSE (↓)	superconduct	0.309	0.305	0.290	0.340	0.341	0.326	0.346	0.345	0.318	0.310
NLL (↓)	superconduct	–	−0.242	3.045	−0.558	0.622	−1.340	−1.169	−0.065	−0.341	−0.954
ECE (↓)	superconduct	–	0.346	0.515	0.151	0.181	0.204	0.243	0.381	0.916	0.183
WS (↓)	superconduct	–	0.451	1.126	0.204	0.250	0.129	0.141	0.569	0.487	0.161
RMSE (↓)	protein	0.612	0.615	0.575	0.723	0.696	0.701	0.666	0.654	0.610	0.610
NLL (↓)	protein	–	0.560	4.314	0.117	1.515	−0.132	−0.080	3.149	0.141	0.073
ECE (↓)	protein	–	0.440	0.649	0.225	0.567	0.288	0.325	0.809	0.565	0.265
WS (↓)	protein	–	0.592	1.379	0.156	8.242	0.182	0.183	1.483	0.336	0.245
RMSE (↓)	year	0.825	0.800	0.765	0.789	0.803	0.775	0.785	0.785	0.786	0.812
NLL (↓)	year	–	11.250	12.064	0.169	1.861	−0.023	0.940	7.952	1.148	0.477
ECE (↓)	year	–	1.151	1.003	0.249	0.230	0.261	1.043	1.126	1.219	0.369
WS (↓)	year	–	2.848	2.543	0.187	0.163	0.164	0.539	2.461	0.625	0.454

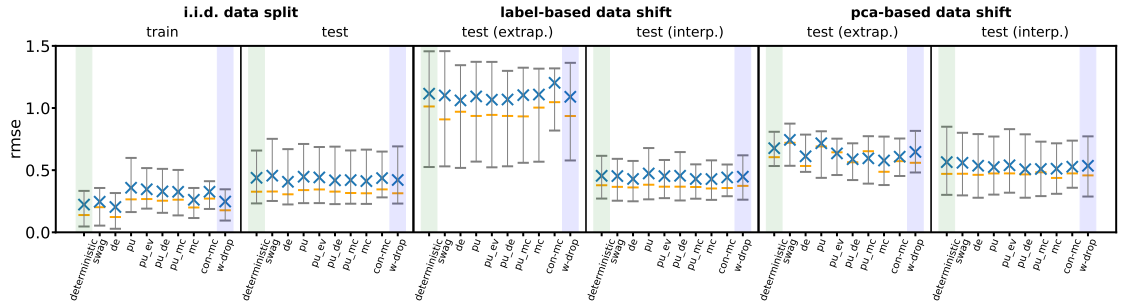


Fig. B.6.: Root-mean-square errors (RMSEs (↓)) of different network types under i.i.d. conditions (first and second panel) and under various kinds of data shift (third to sixth panel). W-dropout (light blue background) is compared to 9 benchmark approaches, including a deterministic model (light green background). Each blue cross is the mean over 14 1D regression datasets. Orange line markers indicate median values. The gray vertical bars reach from the 25% quantile (bottom horizontal line) to the 75% quantile (top horizontal line).

note that the deterministic SqueezeDet yields OD scores that are $5-15\%$ better than the ones of MC- and W-SqueezeDet. This finding equally concerns mAP and the regression scores of mIoU and RMSE. The OD capabilities of the two probabilistic networks are comparable, see also Fig. B.9. It shows that performance losses are less caused by our specific version of dropout but rather generally by using dropout-based techniques.

This performance gap between deterministic and probabilistic models can be understood when recalling that the anchor-based object proposals of SqueezeDet are piped through a multi-step post-processing to obtain the final detections. Dropout-enhanced models, in particular, require an additional step of clustering the stochastic proposals, which may in some cases cause incorrect cluster assignments. Moreover, fewer resulting proposals can be matched with ground truth objects (see the precision values in Tab. B.4). This might be attributed to the stochastic nature of the bounding boxes, which leads to slightly increased errors and therefore, and in combination with the IoU threshold, to less matched proposals. Early experiments suggested that changes to the post-processing routines can contribute to mitigating large parts of these performance losses. As demonstrated in Fig. B.9, we believe that the questions of performance and W-dropout are detached and therefore relegate a deeper exploration of proposal matching to future work.

Further results on SqueezeDet Coordinate-wise regression results and uncertainty scores for MC-SqueezeDet and W-SqueezeDet on KITTI are shown in Tab. B.7. While we observe noteworthy differences between coordinates, the relative ordering of MC-SqueezeDet and W-SqueezeDet for a given measure remains the same.

Analyzing in-data and out-of-data NLL and WS values for all six datasets (see Fig. B.11), we find results that qualitatively resemble those on ECE in Fig. 5.9. W-SqueezeDet

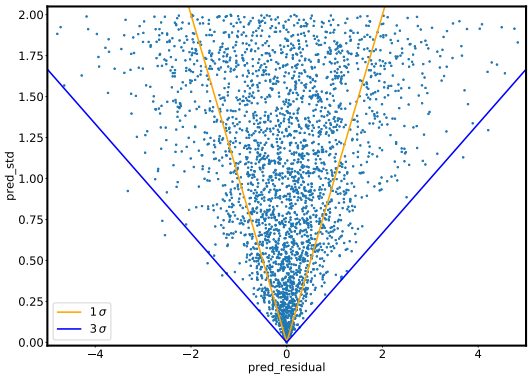


Fig. B.7.: Prediction residuals (x-axis) and predictive uncertainty (y-axis) for a *hypothetical* ideal uncertainty mechanism. The Gaussian errors are matched by Gaussian uncertainty predictions at the exact same scale. 68.3% of all uncertainty estimates (plot points) lie above the orange 1σ-lines and 99.7% of them above the blue 3σ-lines.

outperforms MC-SqueezeDet on the respective i.i.d. test set and also under data shift. For both uncertainty approaches, some NLL values are affected by outliers.

Finally, Fig. B.12 visualizes how various regression and uncertainty (test) scores evolve during model training on the BDD100k dataset. MC-SqueezeDet (dashed) and W-SqueezeDet (solid) "converge" with comparable speed (no changes to test RMSE and mIoU after 100,000 training steps) and reach similar final performances. W-SqueezeDet's explicit optimization of uncertainty estimates yields larger standard deviations (center panel) and smaller values for NLL, ECE, WS and ETL compared to MC-SqueezeDet (center right panel, bottom row). For the unbounded scores NLL, WS and ETL, W-SqueezeDet exhibits higher stability during training.

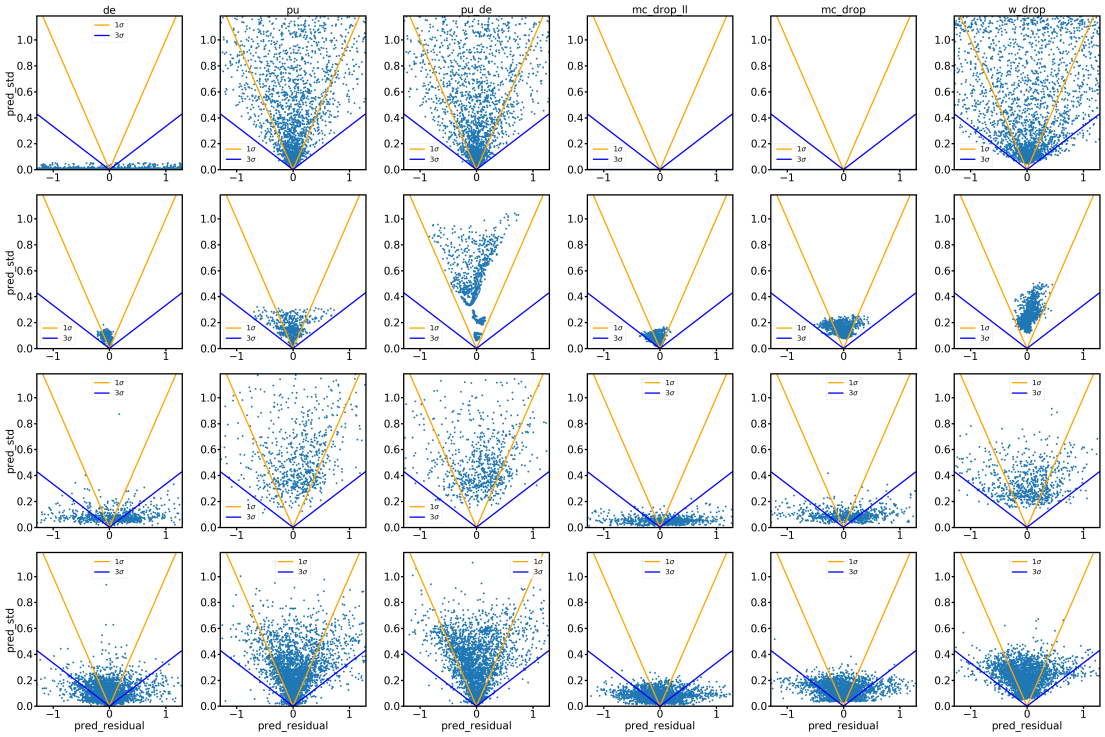


Fig. B.8.: Prediction residuals (respective x-axis) and predictive uncertainty (respective y-axis) for different uncertainty mechanisms (columns) and datasets (rows). Each light blue dot in each plot corresponds to one test data point. Realistic uncertainty estimates should lie mostly above the blue 3σ-lines. The datasets "toy-noise", "naval", "abalone" and "superconduct" are shown, from top to bottom.

Tab. B.4.: Regression performance and uncertainty quality of SqueezeDet-type networks on KITTI test data. W-SqueezeDet (W-SqzDet) is compared with MC-SqueezeDet (MC-SqzDet) and the deterministic network (SqzDet). Extending our set of measures, we additionally report mean average precision (mAP) as well as (class-averaged) recall and precision.

measure	SqzDet	MC-SqzDet	W-SqzDet
mAP (↑)	**0.692**	0.618	0.619
recall (↑)	**0.758**	0.686	0.688
precision (↑)	**0.324**	0.296	0.277
mIoU (↑)	**0.730**	0.695	0.694
RMSE (↓)	**13.058**	14.666	14.505
NLL (↓)	-	25.704	**6.309**
ECE (↓)	-	0.825	**0.433**
WS (↓)	-	2.831	**0.900**
$ETL_{0.99}$	-	42.101	18.223

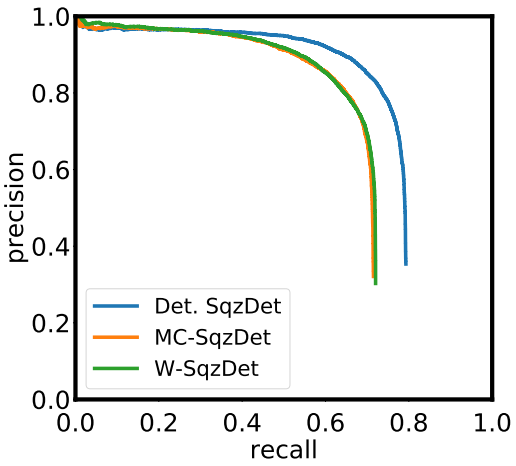


Fig. B.9.: Recall-precision curves of the deterministic SqueezeDet (blue), MC-SqueezeDet (orange) and W-SqueezeDet (green) for the object class "vehicle" in the KITTI test dataset. The AP values of the respective networks (for class "vehicle") are given by the areas under the curves.

Tab. B.5.: General information on the object detection datasets.

dataset	place of data collection	type	train/test split
KITTI	metropolitan area of Karlsruhe	real	semi-custom (seq.-based)
SynScapes	simulation (only urban scenes)	synthetic	custom (random split)
A2D2	highways and cities in Germany	real	custom (seq.-based)
Nightowls	several cities across Europe	real	pre-defined
NuImages	Boston and 3 diverse areas of Singapore	real	pre-defined
BDD100k	New York, San Francisco Bay, Berkeley	real	pre-defined

Tab. B.6.: Harmonization of the object detection datasets. The various object classes of the six object detection datasets (rows) are grouped into the three main categories "vehicle", "pedestrian" and "cyclist" (columns). Some classes are too rare or irregular and are thus discarded.

dataset	vehicle	cyclist	pedestrian	discarded
KITTI	car	cyclist	pedestrian	van, truck, tram, person-sitting, misc, do-not-care
SynScapes	car, motorbike, truck, bus	cyclist	pedestrian	train
A2D2	car, truck	cyclist	pedestrian	-
Nightowls	motorbike	cyclist	pedestrian	ignore-area
NuImages	car, motorbike, truck, vehicle-other	cyclist	pedestrian	movable-object
BDD100k	car, motorbike, truck, bus, trailer, vehicle-other	cyclist	pedestrian, other-person	train, rider, traffic-light, traffic-sign

Fig. B.10.: Two exemplary object detection images from BDD100k (top row, real-world image) and SynScapes (bottom row, synthetic image), respectively. For each original image (left column), two corrupted versions are generated: a blurred one (middle column) and a noisy one (right column), see the text in Section 5.4 for details.

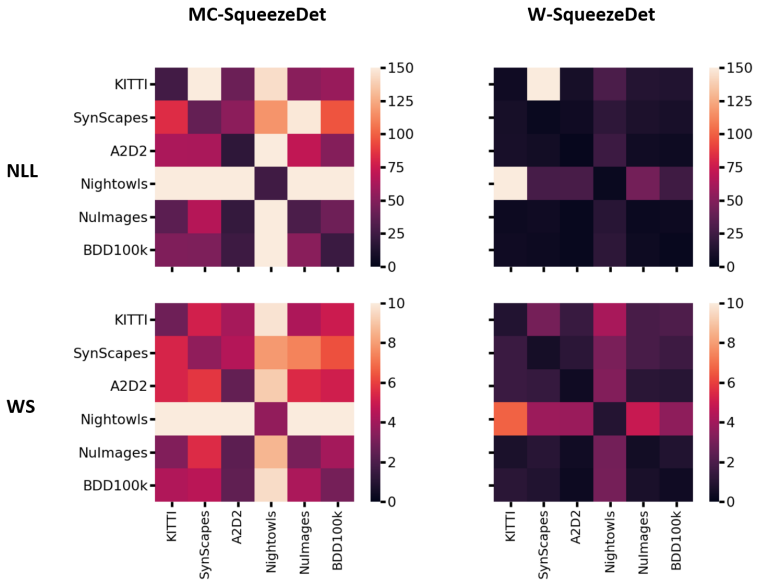


Fig. B.11.: In-data and out-of-data evaluation of MC-SqueezeDet (l.h.s.) and W-SqueezeDet (r.h.s.) on six OD datasets. We consider the negative log-likelihood (NLL, top row) and the Wasserstein measure (WS, bottom row). For each heatmap entry, the row label refers to the training dataset and the column label to the test dataset. Thus, diagonal matrix elements are in-data evaluations, non-diagonal elements are OOD analyses.

Tab. B.7.: Regression performance and uncertainty quality of SqueezeDet-type networks on KITTI train/test data. W-SqueezeDet is compared with MC-SqueezeDet.

	train		**test**	
measure	MC-SqzDet	W-SqzDet	MC-SqzDet	W-SqzDet
mIoU (↑)	**0.705**	0.691	**0.695**	0.694
RMSE (↓)	**8.769**	9.832	14.666	**14.505**
NLL_x (↓)	14.793	**2.808**	34.827	**6.941**
NLL_y	6.135	**2.170**	13.364	**3.808**
NLL_w	6.916	**3.305**	36.384	**8.579**
NLL_h	6.146	**2.796**	18.241	**5.908**
ECE_x (↓)	0.560	**0.148**	0.748	**0.330**
ECE_y	0.659	**0.180**	0.835	**0.419**
ECE_w	0.523	**0.147**	0.888	**0.520**
ECE_h	0.716	**0.296**	0.83	**0.465**
WS_x (↓)	1.729	**0.283**	3.06	**0.830**
WS_y	1.370	**0.299**	2.260	**0.680**
WS_w	1.145	**0.243**	3.485	**1.203**
WS_h	1.442	**0.437**	2.517	**0.888**
$ETL_{0.99,x}$	34.443	**9.316**	55.772	**21.310**
$ETL_{0.99,y}$	18.202	**7.677**	26.675	**11.772**
$ETL_{0.99,w}$	19.914	**10.835**	53.408	**23.202**
$ETL_{0.99,h}$	16.872	**7.584**	32.547	**16.608**

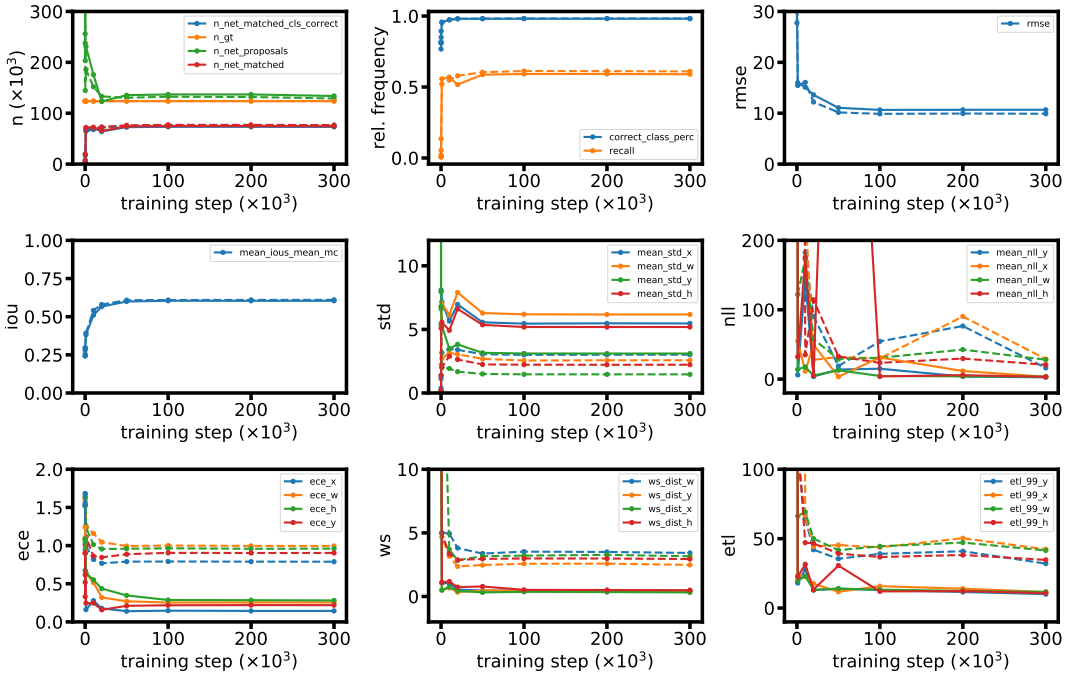


Fig. B.12.: Various test statistics of W-SqueezeDet (solid lines) and MC-SqueezeDet (dashed lines) during model optimization on the BDD100k dataset. We consider performance scores (recall, RMSE, IoU, see first and second row) and uncertainty measures (NLL, ECE, WS, ETL, see second and third row). W-SqueezeDet yields comparable task performance while providing clearly better uncertainty estimates.

B.3. Stability with respect to hyperparameters p and L

W-dropout possesses two hyperparameters: the neuron drop rate p and the sample size L used to calculate the empirical estimates $\mu_{\tilde{\boldsymbol{\theta}}}(x_i)$ and $\sigma_{\tilde{\boldsymbol{\theta}}}(x_i)$. Here, we analyze the impact of these parameters on the quality of accordingly trained models.

For $p = 0.05, 0.1, 0.2, 0.3, 0.4, 0.5$, we observe only relatively small differences in both RMSE (see top panel of Fig. B.13) and ECE (see bottom panel of Fig. B.13). On train data, RMSE slightly deteriorates with increasing p, i.e., with decreasing complexity of the sub-networks. For ECE, we find minor improvements with growing drop rate, which might be explained by the fact that the L sub-networks in a given optimization step overlap less for higher p-values, thus allowing them to approximate the actual data distribution more closely. We choose $p = 0.1$ as the complexity of the resulting sub-networks is only mildly reduced compared to the deterministic full network.

Studying the impact of sample size $L = 4, 5, 8, 10, 20$, we find RMSE (see top panel of Fig. B.14) to be largely stable w.r.t. this parameter. For ECE (see bottom panel of Fig. B.14), train scores grow with L, indicating a certain over-estimation of the present aleatoric uncertainties. This artifact is not generalized to test data though, where we observe broadly similar mean values and 75% quantiles. Under data shift, certain fluctuations of ECE occur as sample size L changes, however, there is no clear trend. We thus choose the rather small $L = 5$ to keep the computational overhead down.

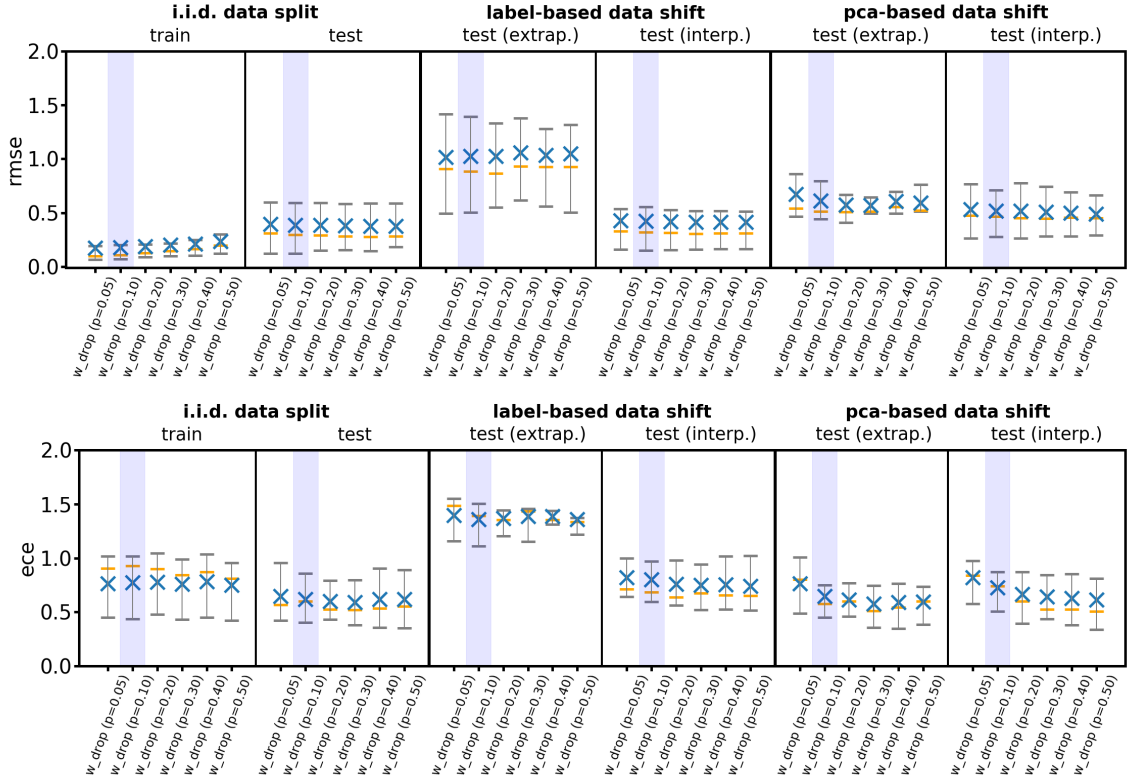


Fig. B.13.: Dependence of Wasserstein dropout on drop rate p. Root-mean-square errors (RMSEs (↓), top row) and expected calibration errors (ECEs (↓), bottom row) are shown for neuron drop rates of $p = 0.05, 0.1, 0.2, 0.3, 0.4, 0.5$ under i.i.d. conditions (first and second panel in each row) and under various kinds of data shift (third to sixth panel in each row, see the text in Subsection 5.3.3 for details). W-dropout with $p = 0.1$ (used for evaluations on toy and 1D regression data) is highlighted by a light blue background. Each blue cross is the mean over 10 standard regression datasets. Orange line markers indicate median values. The gray vertical bars reach from the 25% quantile (bottom horizontal line) to the 75% quantile (top horizontal line).

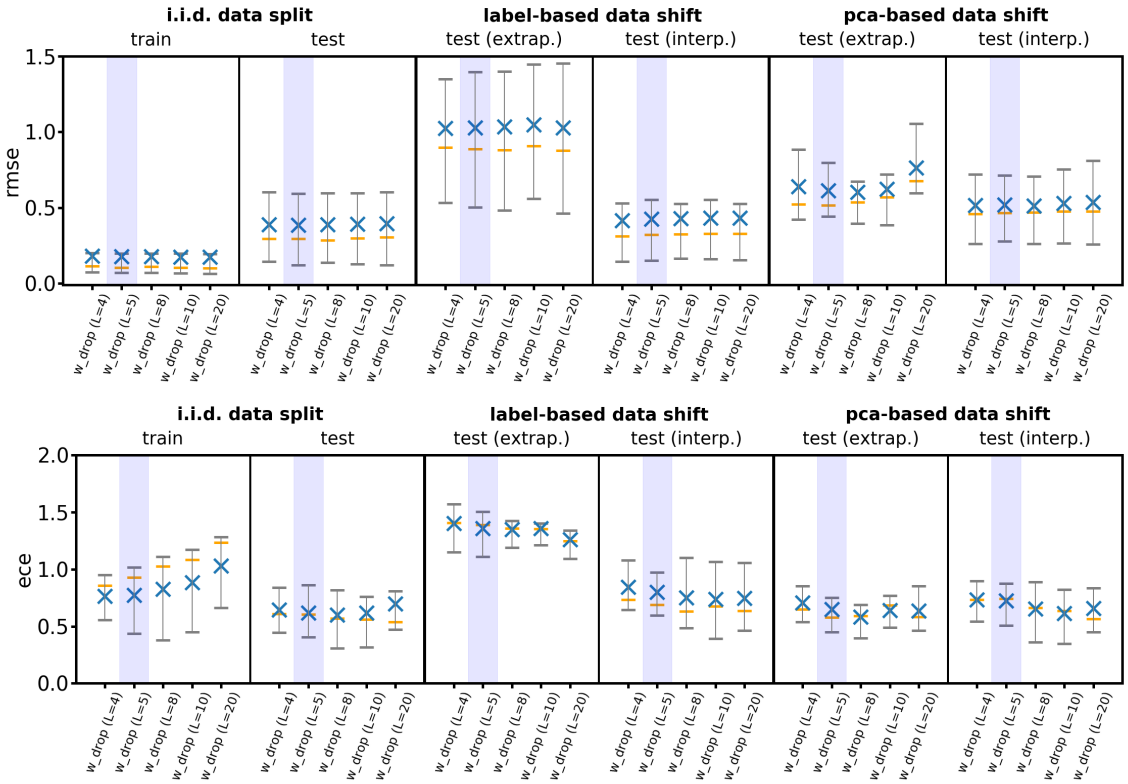


Fig. B.14.: Dependence of Wasserstein dropout on sample size L. Root-mean-square errors (RMSEs (↓), top row) and expected calibration errors (ECEs (↓), bottom row) are shown for sample sizes of $L = 4, 5, 8, 10, 20$ under i.i.d. conditions (first and second panel in each row) and under various kinds of data shift (third to sixth panel in each row, see the text in Subsection 5.3.3 for details). W-dropout with $L = 5$ (used throughout the remainder of the chapter) is highlighted by a light blue background. Each blue cross is the mean over 10 standard regression datasets. Orange line markers indicate median values. The gray vertical bars reach from the 25% quantile (bottom horizontal line) to the 75% quantile (top horizontal line).

B.4. In-depth investigation of uncertainty measures

In the following, we employ the Kolmogorov-Smirnov distance as a supplementary uncertainty score and compare it with expected calibration error (ECE) and Wasserstein distance (WS). Finally, limitations of negative log-likelihood (NLL) for uncertainty quantification are discussed.

B.4.1. Dependencies between uncertainty measures

Extending the analysis of empirically observed dependencies between WS and ECE in Fig. 5.3, we additionally consider Kolmogorov-Smirnov (KS) distances (Stephens, 1974) in Fig. B.15 (middle and bottom panel). These KS-distances are calculated between samples of normalized residuals and a standard Gaussian. Different from the Wasserstein distance, the KS-distance is not transport-based but determined by the largest distance between the empirical CDFs of the two samples. It is therefore bounded to $[0, 1]$ and unable to resolve differences between two samples that both strongly deviate from a standard Gaussian. Again, we find the dependencies between these measures to clearly deviate from ideal correlation.

The data splits in Fig. 5.3 and Fig. B.15 are color-coded as follows: train is green, test is blue, PCA-interpolate is green-yellow, PCA-extrapolate is orange-yellow, label-interpolate is red and label-extrapolate is light red. The mapping between uncertainty methods and plot markers reads: SWAG is “triangle”, MC is “diamond”, MC-LL is “thin diamond”, DE is “cross”, PU is “point”, PU-DE is “star”, PU-MC is “circle”, PU-EV is “pentagon” and W-dropout is “plus”. The data base of this visualization are the 14 standard regression datasets. Some Wasserstein distances lie above the x-axis cut-off and are thus not visualized.

B.4.2. Discussion of NLL as a measure of uncertainty

Typically, DNNs using uncertainty are often evaluated in terms of their negative log-likelihood (NLL). This property is affected not only by the uncertainty, but also by the DNNs performance. Additionally, it is difficult to interpret, sometimes leading to counterintuitive results, which we want to elaborate on here. As a first example, take the likelihood of two datasets $x_1 = \{0\}$ and $x_2 = \{0.5\}$, each consisting of a single point, with respect to a normal distribution $\mathcal{N}(0, 1)$. Naturally, we find x_1 to be located at the maximum of the considered normal distribution and deem it the more likely candidate. But, if we extend these datasets to more than single points, i.e., $\tilde{x}_1 = \{0, 0.1, 0, -0.1, 0\}$ and $\tilde{x}_2 = \{0.5, -0.4, 0, -1.9, -0.7\}$, it becomes obvious that $\tilde{x}_2$ is much more likely to follow the intended Gaussian distribution. Nonetheless, $\mathrm{NLL}(\tilde{x}_2) \approx 1.4 > 0.9 \approx \mathrm{NLL}(\tilde{x}_1)$, where

$$\mathrm{NLL}(y) := \log \sqrt{2\pi\sigma^2} + \frac{1}{N} \sum_{i=1}^{N} \frac{(y_i - \mu)^2}{2\sigma^2} \quad . \tag{B.3}$$

This may be seen as a direct consequence of the pointwise definition of NLL, which does not consider the distribution of the elements in $\tilde{x}_i$. From this observation also follows that a model with high prediction accuracy will have a lower NLL score as a worse performing one if uncertainties are predicted in the same way. Independent of whether those reflected the "true" uncertainty in either case. This issue can be further substantiated on a second example. Consider two other datasets z_1, z_2 drawn i.i.d. from Gaussian distributions $\mathcal{N}(0, \sigma_i)$ with two differing values $\sigma_1 < \sigma_2$. If we determine the NLL of each with respect to its own distribution the offset term in Eq. B.3 leads to $\text{NLL}(z_2) = \text{NLL}(z_1) + \log(\sigma_2/\sigma_1)$ with $\log(\sigma_2/\sigma_1) > 0$. Although both accurately reflect their own distributions, or uncertainties so to speak, the narrower z_1 is more "likely". This offset makes it difficult to assess reported NLL values for systems with heteroscedastic uncertainty. While smaller is typically "better", it is highly data- (and prediction-)dependent which value is good in the sense of a reasonable correlation between performance and uncertainty.

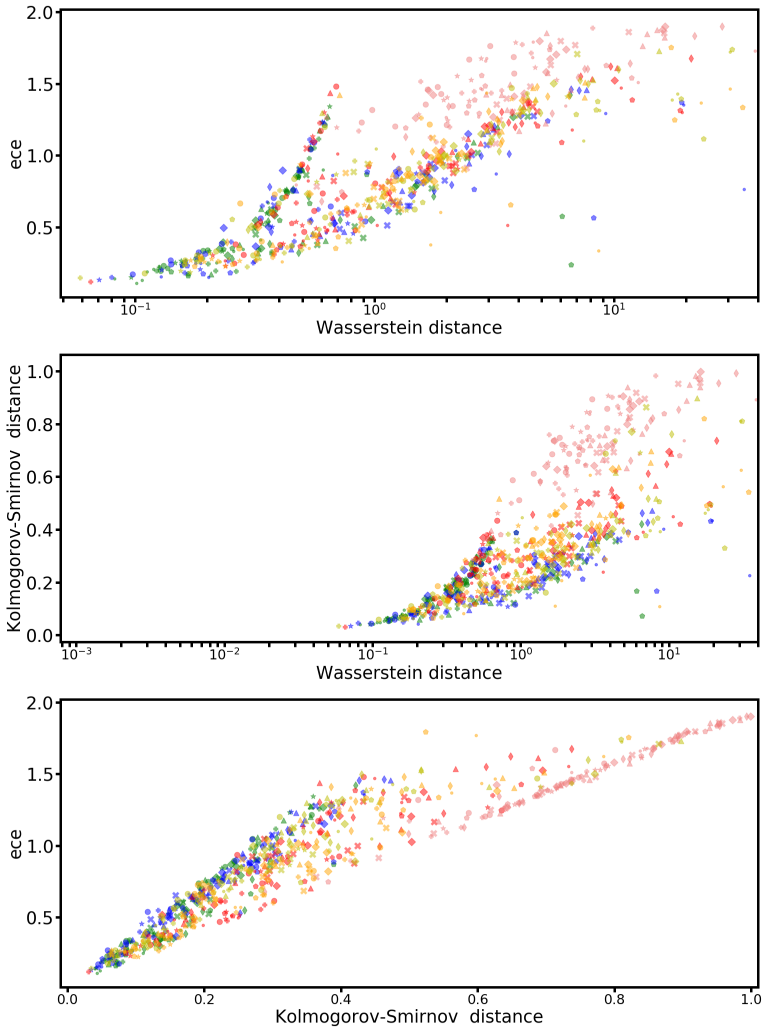


Fig. B.15.: Dependencies between the three uncertainty measures ECE, Wasserstein distance and Kolmogorov-Smirnov distance. Uncertainty methods are encoded via plot markers, data splits via color. Datasets are not encoded and cannot be distinguished (see the text in Subsection B.4.1 for more details). Each plot point corresponds to a cross-validated trained network. The clearly visible deviations from ideal correlations point at the potential of these uncertainty measures to complement one another.

List of Acronyms

AD	Autonomous driving
BNN	Bayesian neural network
CDF	Cumulative distribution function
CLT	Central limit theorem
CNN	Convolutional neural network
DE	Deep ensemble
DL	Deep learning
DNN	Deep neural network
DOF	Degree of freedom
ECE	Expected calibration error
ELBO	Evidence lower bound
EM	Expectation-maximization
ETL	Expected tail loss
FCN	Fully connected network
GD	Gradient descent
GL	Gaussian likelihood
GL-OS	Gaussian-likelihood one-sample
GP	Gaussian process
HMM	Hidden Markov model
ID	In-distribution
IoU	Intersection over union
MAD	Mean absolute deviation
MAP	Maximum a posteriori
MC	Monte Carlo
MCMC	Markov chain Monte Carlo
ML	Machine learning
MLE	Maximum likelihood estimation
MLP	Multilayer perceptron

MSE	Mean squared error
NLL	Negative log-likelihood
NMF	Non-negative matrix factorization
NMS	Non-maximum suppression
NN	Neural network
OD	Object detection
ODD	Operational design domain
OOD	Out-of-distribution
PDF	Probability density function
PU	Parametric uncertainty
RMSE	Root-mean-square error
SGD	Stochastic gradient descent
VI	Variational inference
VRU	Vulnerable road user
WS	Wasserstein distance

List of Figures

List of Tables

Bibliography

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In Kimberly Keeton and Timothy Roscoe (eds.), *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016*, pp. 265–283. USENIX Association, 2016. URL https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi. 35, 138

Stephanie Abrecht, Lydia Gauerhof, Christoph Gladisch, Konrad Groh, Christian Heinzemann, and Matthias Woehrle. Testing deep learning-based visual perception for automated driving. *ACM Trans. Cyber Phys. Syst.*, 5(4):37:1–37:28, 2021. doi: 10.1145/3450356. URL https://doi.org/10.1145/3450356. 95

Tameem Adel, Isabel Valera, Zoubin Ghahramani, and Adrian Weller. One-network adversarial fairness. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 2412–2420. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33012412. URL https://doi.org/10.1609/aaai.v33i01.33012412. 95

Ahmed Alaa and Mihaela Van Der Schaar. Discriminative jackknife: Quantifying uncertainty in deep learning via higher-order influence functions. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 165–174. PMLR, 7 2020. URL http://proceedings.mlr.press/v119/alaa20a.html. 38

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. *CoRR*, abs/2204.14198, 2022. doi: 10.48550/arXiv.2204.14198. URL https://doi.org/10.48550/arXiv.2204.14198. 132

Felipe Almeida and Geraldo Xexéo. Word embeddings: A survey. *CoRR, abs/1901.09069*, 2019. 42

Alexander Amini, Wilko Schwarting, Ava Soleimany, and Daniela Rus. Deep evidential regression. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 14927–14937, 2020. URL `https://proceedings.neurips.cc/paper/2020/file/aab085461de182608ee9f607f3f7d18f-Paper.pdf`. 4, 31, 72, 112

Idan Amir, Tomer Koren, and Roi Livni. SGD generalizes better than GD (and regularization doesn't help). In Mikhail Belkin and Samory Kpotufe (eds.), *Conference on Learning Theory, COLT 2021, 15-19 August 2021, Boulder, Colorado, USA*, volume 134 of *Proceedings of Machine Learning Research*, pp. 63–92. PMLR, 2021. URL `http://proceedings.mlr.press/v134/amir21a.html`. 19

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *CoRR*, abs/1606.06565, 2016. URL `http://arxiv.org/abs/1606.06565`. 1, 43

Animashree Anandkumar, Daniel J. Hsu, and Sham M. Kakade. A method of moments for mixture models and hidden Markov models. In Shie Mannor, Nathan Srebro, and Robert C. Williamson (eds.), *COLT 2012 - The 25th Annual Conference on Learning Theory, June 25-27, 2012, Edinburgh, Scotland*, volume 23 of *JMLR Proceedings*, pp. 33.1–33.34. JMLR.org, 2012. URL `http://proceedings.mlr.press/v23/anandkumar12/anandkumar12.pdf`. 44

Vincent Aravantinos and Peter Schlicht. Making the relationship between uncertainty estimation and safety less uncertain. In *2020 Design, Automation & Test in Europe Conference & Exhibition, DATE 2020, Grenoble, France, March 9-13, 2020*, pp. 1139–1144. IEEE, 2020. doi: 10.23919/DATE48585.2020.9116541. URL `https://doi.org/10.23919/DATE48585.2020.9116541`. 91

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 214–223. PMLR, 8 2017. URL `https://proceedings.mlr.press/v70/arjovsky17a.html`. 22, 69

Samuel G Armato III, Geoffrey McLennan, Luc Bidaut, Michael F McNitt-Gray, Charles R Meyer, Anthony P Reeves, Binsheng Zhao, Denise R Aberle, Claudia I Henschke, Eric A Hoffman, et al. The lung image database consortium (LIDC) and image database resource initiative (IDRI): a completed reference database of lung nodules on CT scans. *Medical physics*, 38(2):915–931, 2011. 3

Matthew Arnold, Rachel K. E. Bellamy, Michael Hind, Stephanie Houde, Sameep Mehta, Aleksandra Mojsilovic, Ravi Nair, Karthikeyan Natesan Ramamurthy, Alexandra

Olteanu, David Piorkowski, Darrell Reimer, John T. Richards, Jason Tsay, and Kush R. Varshney. Factsheets: Increasing trust in AI services through supplier's declarations of conformity. *IBM J. Res. Dev.*, 63(4/5):6:1–6:13, 2019. doi: 10.1147/JRD.2019.2942288. URL `https://doi.org/10.1147/JRD.2019.2942288`. 95, 130

Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artif. Intell.*, 297:103500, 2021. doi: 10.1016/j.artint.2021.103500. URL `https://doi.org/10.1016/j.artint.2021.103500`. 131

Ehsaneddin Asgari and Mohammad RK Mofrad. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS One*, 10(11):e0141287, 2015. 42

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1409.0473`. 42

Lalit Bahl, Peter Brown, Peter De Souza, and Robert Mercer. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *ICASSP'86. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 11, pp. 49–52. IEEE, 1986. 42

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018. 44

Leonardo Baldassini and Jose Antonio Rodríguez Serrano. client2vec: towards systematic baselines for banking applications. *arXiv preprint arXiv:1802.04198*, 2018. 42

Pierre Baldi and Yves Chauvin. Smooth on-line learning algorithms for hidden Markov models. *Neural Computation*, 6(2):307–318, 1994. 44

Wentao Bao, Qi Yu, and Yu Kong. Uncertainty-based traffic accident anticipation with spatio-temporal relational learning. In Chang Wen Chen, Rita Cucchiara, Xian-Sheng Hua, Guo-Jun Qi, Elisa Ricci, Zhengyou Zhang, and Roger Zimmermann (eds.), *MM '20: The 28th ACM International Conference on Multimedia, Virtual Event / Seattle, WA, USA, October 12-16, 2020*, pp. 2682–2690. ACM, 2020. doi: 10.1145/3394171.3413827. URL `https://doi.org/10.1145/3394171.3413827`. 121

Rina Foygel Barber, Emmanuel J. Candès, Aaditya Ramdas, and Ryan J. Tibshirani. Predictive inference with the jackknife+. *The Annals of Statistics*, 49(1):486 – 507, 2021. doi: 10.1214/20-AOS1965. URL `https://doi.org/10.1214/20-AOS1965`. 4, 38

Jonathan T. Barron. A general and adaptive robust loss function. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA,*

June 16-20, 2019, pp. 4331–4339. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00446. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Barron_A_General_and_Adaptive_Robust_Loss_Function_CVPR_2019_paper.html. 18

Sue Becker and Yann LeCun. Improving the convergence of back-propagation learning with second order methods. In David S. Touretzky, Geoffrey E. Hinton, and Terrence J. Sejnowski (eds.), *Proceedings of the 1988 Connectionist Models Summer School*, pp. 29–37. San Francisco, CA: Morgan Kaufmann, 1989. 33

Tristan Bepler and Bonnie Berger. Learning protein sequence embeddings using information from structure. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019. 53

Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, Talapady N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The protein data bank. *Nucleic acids research*, 28(1):235–242, 2000. 50, 51

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738. 3, 20, 21, 32, 33, 36, 37, 41, 46, 84

David M. Blei and Michael I. Jordan. Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1(1):121 – 143, 2006. doi: 10.1214/06-BA104. URL https://doi.org/10.1214/06-BA104. 39

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017. 21

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1613–1622, Lille, France, 7 2015. PMLR. URL https://proceedings.mlr.press/v37/blundell15.html. 33

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 1

Léon Bottou. *On-Line Learning and Stochastic Approximations*, pp. 9–42. Cambridge University Press, USA, 1999. ISBN 0521652634. 19

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010. 48

Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Rev.*, 60(2):223–311, 2018. doi: 10.1137/16M1080173. URL https://doi.org/10.1137/16M1080173. 19

Hervé Bourlard, Nelson Morgan, and Steve Renals. Neural nets and hidden Markov models: Review and generalizations. *Speech Communication*, 11(2-3):237–246, 1992. 44

Houssem Ben Braiek and Foutse Khomh. On testing machine learning programs. *J. Syst. Softw.*, 164:110542, 2020. doi: 10.1016/j.jss.2020.110542. URL https://doi.org/10.1016/j.jss.2020.110542. 96

Martin Brandt, Compton J Tucker, Ankit Kariryaa, Kjeld Rasmussen, Christin Abel, Jennifer Small, Jerome Chave, Laura Vang Rasmussen, Pierre Hiernaux, Abdoul Aziz Diouf, et al. An unexpectedly large count of trees in the West African Sahara and Sahel. *Nature*, 587(7832):78–82, 2020. 82

Wieland Brendel and Matthias Bethge. Approximating CNNs with bag-of-local-features models works surprisingly well on ImageNet. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=SkfMWhAqYQ. 95

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html. 132

Miles Brundage, Shahar Avin, Jasmine Wang, Haydn Belfield, Gretchen Krueger, Gillian Hadfield, Heidy Khlaaf, Jingying Yang, Helen Toner, Ruth Fong, et al. Toward trustworthy AI development: Mechanisms for supporting verifiable claims. *arXiv preprint arXiv:2004.07213*, 2020. 43

David R. Burt, Carl Edward Rasmussen, and Mark van der Wilk. Rates of convergence for sparse variational gaussian process regression. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 862–871. PMLR, 2019. URL http://proceedings.mlr.press/v97/burt19a.html. 34

Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 11618–11628. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.01164. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Caesar_nuScenes_A_Multimodal_Dataset_for_Autonomous_Driving_CVPR_2020_paper.html. 84

Yuanqiang Cai, Longyin Wen, Libo Zhang, Dawei Du, and Weiqiang Wang. Rethinking object detection in retail stores. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 947–954. AAAI Press, 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/16178. 82

Ricardo J. G. B. Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu (eds.), *Advances in Knowledge Discovery and Data Mining, 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, April 14-17, 2013, Proceedings, Part II*, volume 7819 of *Lecture Notes in Computer Science*, pp. 160–172. Springer, 2013. doi: 10.1007/978-3-642-37456-2_14. URL https://doi.org/10.1007/978-3-642-37456-2_14. 84

Challenging Traffic Scenes, 2022. top left: https://commons.m.wikimedia.org/wiki/File:Manifestacio_en_Erevano_la_23-an_de_novembro_2021_%2801%29_10.jpg, top middle: https://commons.wikimedia.org/wiki/File:Jersey_Town_Criterium_2010_recumbent_056.jpg, top right: https://static.independent.co.uk/s3fs-public/thumbnails/image/2015/08/03/18/minioncrop.jpg, with kind permission of Erin Van Londen, bottom left: https://commons.wikimedia.org/wiki/File:Harajuku_-_pedestrians_on_Omotesando_07_%2815716216586%29.jpg, bottom middle: https://commons.wikimedia.org/wiki/File:Road_for_Cow_-_panoramio.jpg, bottom right: https://picryl.com/media/members-of-the-combined-allied-parachute-demonstration-team-land-on-17th-of-141be9 (all accessed on 2022-07-06). 2

Fu-Hsiang Chan, Yu-Ting Chen, Yu Xiang, and Min Sun. Anticipating accidents in dashcam videos. In Shang-Hong Lai, Vincent Lepetit, Ko Nishino, and Yoichi Sato (eds.), *Computer Vision - ACCV 2016 - 13th Asian Conference on Computer Vision, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part IV*, volume 10114 of *Lecture Notes in Computer Science*, pp. 136–153. Springer, 2016. doi: 10.1007/978-3-319-54190-7_9. URL https://doi.org/10.1007/978-3-319-54190-7_9. 121

Robin Chan, Matthias Rottmann, Fabian Hüger, Peter Schlicht, and Hanno Gottschalk. Application of decision rules for handling class imbalance in semantic segmentation. *CoRR*, abs/1901.08394, 2019. URL `http://arxiv.org/abs/1901.08394`. 113

Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann. Posterior network: Uncertainty estimation without OOD samples via density-based pseudo-counts. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/0eac690d7059a8de4b48e90f14510391-Abstract.html`. 111

Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, 1999. 42

Tianqi Chen, Emily B. Fox, and Carlos Guestrin. Stochastic gradient Hamiltonian Monte Carlo. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pp. 1683–1691. JMLR.org, 2014. URL `http://proceedings.mlr.press/v32/cheni14.html`. 36

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 2020. URL `http://proceedings.mlr.press/v119/chen20j.html`. 11, 13

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 1800–1807. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.195. URL `https://doi.org/10.1109/CVPR.2017.195`. 35

K Alec Chrystal, Paul D Mizen, and PD Mizen. Goodhart's law: its origins, meaning and implications for monetary policy. *Central banking, monetary theory and practice: Essays in honour of Charles Goodhart*, 1:221–243, 2003. 108

Youngseog Chung, Ian Char, Han Guo, Jeff Schneider, and Willie Neiswanger. Uncertainty toolbox: an open-source library for assessing, visualizing, and improving uncertainty quantification. *CoRR*, abs/2109.10254, 2021a. URL `https://arxiv.org/abs/2109.10254`. 96, 125

Youngseog Chung, Willie Neiswanger, Ian Char, and Jeff Schneider. Beyond pinball loss: Quantile methods for calibrated uncertainty quantification. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021,*

virtual, pp. 10971–10984, 2021b. URL https://proceedings.neurips.cc/paper/2021/hash/5b168fdba5ee5ea262cc2d4c0b457697-Abstract.html. 32

Anthony Corso, Robert J. Moss, Mark Koren, Ritchie Lee, and Mykel J. Kochenderfer. A survey of algorithms for black-box safety validation of cyber-physical systems. *J. Artif. Intell. Res.*, 72:377–428, 2021. doi: 10.1613/jair.1.12716. URL https://doi.org/10.1613/jair.1.12716. 96

George Cybenko and Valentino Crespi. Learning hidden Markov models using nonnegative matrix factorization. *IEEE Transactions on Information Theory*, 57(6):3963–3970, 2011. 44

Andreas C. Damianou and Neil D. Lawrence. Deep Gaussian processes. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2013, Scottsdale, AZ, USA, April 29 - May 1, 2013*, volume 31 of *JMLR Workshop and Conference Proceedings*, pp. 207–215. JMLR.org, 2013. URL http://proceedings.mlr.press/v31/damianou13a.html. 34

Morris H DeGroot and Stephen E Fienberg. The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 32(1-2):12–22, 1983. 39, 105

Frederik Michel Dekking, Cornelis Kraaikamp, Hendrik Paul Lopuhaä, and Ludolf Erwin Meester. *A Modern Introduction to Probability and Statistics: Understanding why and how*. Springer Science & Business Media, 2005. 63

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977. 42

Li Deng, Jinyu Li, Jui-Ting Huang, Kaisheng Yao, Dong Yu, Frank Seide, Michael Seltzer, Geoff Zweig, Xiaodong He, Jason Williams, et al. Recent advances in deep learning for speech research at Microsoft. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8604–8608. IEEE, 2013. 42

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423. 11, 13, 42

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml. 71

Nikita Durasov, Timur M. Bagautdinov, Pierre Baqué, and Pascal Fua. Masksembles for uncertainty estimation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pp. 13539–13548. Computer Vision Foundation / IEEE, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Durasov_Masksembles_for_Uncertainty_Estimation_CVPR_2021_paper.html. 4, 38

Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pp. 214–226, 2012. 43

Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. Patterns in property specifications for finite-state verification. In Barry W. Boehm, David Garlan, and Jeff Kramer (eds.), *Proceedings of the 1999 International Conference on Software Engineering, ICSE' 99, Los Angeles, CA, USA, May 16-22, 1999*, pp. 411–420. ACM, 1999. doi: 10.1145/302405.302672. URL https://doi.org/10.1145/302405.302672. 92

Marie-Aude Esteve, Joost-Pieter Katoen, Viet Yen Nguyen, Bart Postma, and Yuri Yushtein. Formal correctness, safety, dependability, and performance analysis of a satellite. In Martin Glinz, Gail C. Murphy, and Mauro Pezzè (eds.), *34th International Conference on Software Engineering, ICSE 2012, June 2-9, 2012, Zurich, Switzerland*, pp. 1022–1031. IEEE Computer Society, 2012. doi: 10.1109/ICSE.2012.6227118. URL https://doi.org/10.1109/ICSE.2012.6227118. 92

European Commission. Proposal for a regulation laying down harmonised rules on artificial intelligence, 2021. URL https://digital-strategy.ec.europa.eu/en/library/proposal-regulation-laying-down-harmonised-rules-artificial-intelligence. (accessed on 2021-05-26). 94, 125

Executive Office of the U.S. President. Guidance for regulation of artificial intelligence applications, 2020. URL https://www.whitehouse.gov/wp-content/uploads/2020/11/M-21-06.pdf. (accessed on 2021-05-26). 94

Xinjie Fan, Shujian Zhang, Korawat Tanwisuth, Xiaoning Qian, and Mingyuan Zhou. Contextual dropout: An efficient sample-dependent dropout module. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=ct8_a9h1M. 4

Cong Fang, Zhouchen Lin, and Tong Zhang. Sharp analysis for nonconvex SGD escaping from saddle points. In Alina Beygelzimer and Daniel Hsu (eds.), *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, volume 99 of *Proceedings of Machine Learning Research*, pp. 1192–1234. PMLR, 2019. URL http://proceedings.mlr.press/v99/fang19a.html. 19

Di Feng, Lars Rosenbaum, Claudius Gläser, Fabian Timm, and Klaus Dietmayer. Can we trust you? on calibration of a probabilistic object detector for autonomous driving. *CoRR*, abs/1909.12358, 2019. URL `http://arxiv.org/abs/1909.12358`. 65

Di Feng, Ali Harakeh, Steven L. Waslander, and Klaus Dietmayer. A review and comparative study on probabilistic object detection in autonomous driving. *CoRR*, abs/2011.10671, 2020. URL `https://arxiv.org/abs/2011.10671`. 87, 88

Hans Fischer. *A history of the central limit theorem: From classical to modern probability theory*. Springer, 2011. 20

Andrew YK Foong, Yingzhen Li, José Miguel Hernández-Lobato, and Richard E Turner. 'In-between' uncertainty in Bayesian neural networks. *ICML Workshop on Uncertainty and Robustness in Deep Learning*, 2019. 74

Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *NeurIPS Workshop on Bayesian Deep Learning*, 2019. 38

Nick Foti, Jason Xu, Dillon Laird, and Emily Fox. Stochastic variational inference for hidden Markov models. In *Advances in Neural Information Processing Systems*, pp. 3599–3607, 2014. 44

Rongrong Fu, Hong Wang, and Wenbo Zhao. Dynamic driver fatigue detection using hidden Markov model in real driving condition. *Expert Systems with Applications*, 63: 397–411, 2016. 42

Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1(2):119–130, 1988. doi: 10.1016/0893-6080(88)90014-7. URL `https://doi.org/10.1016/0893-6080(88)90014-7`. 15, 16

Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016. 27

Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with Bernoulli approximate variational inference. *CoRR*, abs/1506.02158, 2015. URL `http://arxiv.org/abs/1506.02158`. 35

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1050–1059, New York, New York, USA, 6 2016a. PMLR. URL `http://proceedings.mlr.press/v48/gal16.html`. 4, 34, 35, 39, 62, 72, 74, 112, 127, 131

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (eds.), *Advances in Neural Information Processing*

Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain, pp. 1019–1027, 2016b. URL https://proceedings.neurips.cc/paper/2016/hash/076a0c97d09cf1a0ec3e19c7f2529f2b-Abstract.html. 35

Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 3581–3590, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/84ddfb34126fc3a48ee38d7044e87276-Abstract.html. 6, 35, 72

Sujan Sai Gannamaneni, Sebastian Houben, and Maram Akila. Semantic concept testing in autonomous driving by extraction of object-level annotations from CARLA. In *IEEE/CVF International Conference on Computer Vision Workshops, ICCVW 2021, Montreal, Canada, October 11-17, 2021*, pp. 1006–1014. IEEE, 2021. doi: 10.1109/ICCVW54120.2021.00117. URL https://doi.org/10.1109/ICCVW54120.2021.00117. 114

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pp. 3354–3361. IEEE Computer Society, 2012. doi: 10.1109/CVPR.2012.6248074. URL https://doi.org/10.1109/CVPR.2012.6248074. 4, 84

Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. Fast approximate natural gradient descent in a Kronecker factored eigenbasis. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 9573–9583, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/48000647b315f6f00f913caa757a70b3-Abstract.html. 33

Jakob Geyer, Yohannes Kassahun, Mentar Mahmudi, Xavier Ricou, Rupesh Durgesh, Andrew S. Chung, Lorenz Hauswald, Viet Hoang Pham, Maximilian Mühlegg, Sebastian Dorn, Tiffany Fernandez, Martin Jänicke, Sudesh Mirashi, Chiragkumar Savani, Martin Sturm, Oleksandr Vorobiov, Martin Oelker, Sebastian Garreis, and Peter Schuberth. A2D2: Audi Autonomous Driving Dataset, 2020. URL https://www.a2d2.audi. 4, 84

Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. Dropblock: A regularization method for convolutional networks. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 10750–10760,

2018. URL https://proceedings.neurips.cc/paper/2018/hash/7edcfb2d8f6a659ef4cd1e6c9b6d7079-Abstract.html. 25

Soumya Ghosh, Q. Vera Liao, Karthikeyan Natesan Ramamurthy, Jirí Navrátil, Prasanna Sattigeri, Kush R. Varshney, and Yunfeng Zhang. Uncertainty quantification 360: A holistic toolkit for quantifying and communicating the uncertainty of AI. *CoRR*, abs/2106.01410, 2021. URL https://arxiv.org/abs/2106.01410. 96, 125

Cinzia Giannetti. A framework for improving process robustness with quantification of uncertainties in industry 4.0. In Piotr Jedrzejowicz, Tülay Yildirim, and Ireneusz Czarnowski (eds.), *IEEE International Conference on INnovations in Intelligent SysTems and Applications, INISTA 2017, Gdynia, Poland, July 3-5, 2017*, pp. 189–194. IEEE, 2017. doi: 10.1109/INISTA.2017.8001155. URL https://doi.org/10.1109/INISTA.2017.8001155. 91

Ross B. Girshick. Fast R-CNN. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pp. 1440–1448. IEEE Computer Society, 2015. doi: 10.1109/ICCV.2015.169. URL https://doi.org/10.1109/ICCV.2015.169. 18

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of Machine Learning Research*, volume 15, pp. 315–323, Fort Lauderdale, FL, USA, 4 2011. JMLR Workshop and Conference Proceedings. 152

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437. URL https://doi.org/10.1198/016214506000001437. 5

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 2672–2680, 2014. URL https://proceedings.neurips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html. 23

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6572. 93

Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016. ISBN 978-0-262-03561-3. URL http://www.deeplearningbook.org/. 42

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander J. Smola. A kernel two-sample test. *J. Mach. Learn. Res.*, 13:723–773, 2012. URL http://dl.acm.org/citation.cfm?id=2188410. 131

Thomas L Griffiths and Zoubin Ghahramani. The Indian buffet process: An introduction and review. *Journal of Machine Learning Research*, 12(Apr):1185–1224, 2011. 44

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 855–864. ACM, 2016. 42

Lars Grunske. Specification patterns for probabilistic quality properties. In Wilhelm Schäfer, Matthew B. Dwyer, and Volker Gruhn (eds.), *30th International Conference on Software Engineering (ICSE 2008), Leipzig, Germany, May 10-18, 2008*, pp. 31–40. ACM, 2008. doi: 10.1145/1368088.1368094. URL https://doi.org/10.1145/1368088.1368094. 92

Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017. 43

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 2017. URL http://proceedings.mlr.press/v70/guo17a.html. 32, 39, 101, 114

Kartik Gupta, Amir Rahimi, Thalaiyasingam Ajanthan, Thomas Mensink, Cristian Sminchisescu, and Richard Hartley. Calibration of neural networks using splines. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=eQe8DEWNN2W. 32, 39

Fredrik K. Gustafsson, Martin Danelljan, and Thomas B. Schön. Evaluating scalable Bayesian deep learning methods for robust computer vision. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*, pp. 1289–1298. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPRW50498.2020.00167. URL https://openaccess.thecvf.com/content_CVPRW_2020/html/w20/Gustafsson_Evaluating_Scalable_Bayesian_Deep_Learning_Methods_for_Robust_Computer_Vision_CVPRW_2020_paper.html. 38

Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J. Russell, and Anca D. Dragan. Inverse reward design. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6765–6774, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/32fdab6559cdfa4f167f8c31b9199643-Abstract.html. 104

David Hall, Feras Dayoub, John Skinner, Haoyang Zhang, Dimity Miller, Peter Corke, Gustavo Carneiro, Anelia Angelova, and Niko Sünderhauf. Probabilistic object detection: Definition and evaluation. In *IEEE Winter Conference on Applications of Computer Vision, WACV 2020, Snowmass Village, CO, USA, March 1-5, 2020*, pp. 1020–1029. IEEE, 2020. doi: 10.1109/WACV45572.2020.9093599. URL https://doi.org/10.1109/WACV45572.2020.9093599. 82, 109

Ali Harakeh, Michael Smart, and Steven L. Waslander. BayesOD: A Bayesian approach for uncertainty estimation in deep object detectors. In *2020 IEEE International Conference on Robotics and Automation, ICRA 2020, Paris, France, May 31 - August 31, 2020*, pp. 87–93. IEEE, 2020. doi: 10.1109/ICRA40945.2020.9196544. URL https://doi.org/10.1109/ICRA40945.2020.9196544. 31, 88

Alan Hartman. Software and hardware testing using combinatorial covering suites. *Journal of Graph Theory - JGT*, 34:237–266, 03 2006. doi: 10.1007/0-387-25036-0_10. 93

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009. 23, 63

W. Keith Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. doi: 10.1093/biomet/57.1.97. URL http://biomet.oxfordjournals.org/cgi/content/abstract/57/1/97. 36

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pp. 1026–1034. IEEE Computer Society, 2015. doi: 10.1109/ICCV.2015.123. URL https://doi.org/10.1109/ICCV.2015.123. 15

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90. 13

Yihui He, Chenchen Zhu, Jianren Wang, Marios Savvides, and Xiangyu Zhang. Bounding box regression with uncertainty for accurate object detection. In *IEEE Conference*

on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019, pp. 2888–2897. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00300. URL http://openaccess.thecvf.com/content_CVPR_2019/html/He_Bounding_Box_Regression_With_Uncertainty_for_Accurate_Object_Detection_CVPR_2019_paper.html. 103

Mikael Henaff, Alfredo Canziani, and Yann LeCun. Model-predictive policy learning with uncertainty regularization for driving in dense traffic. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=HygQBn0cYm. 102

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *CoRR*, abs/1610.02136, 2017. 3, 40

Dan Hendrycks, Steven Basart, Mantas Mazeika, Mohammadreza Mostajabi, Jacob Steinhardt, and Dawn Song. Scaling out-of-distribution detection for real-world settings. *arXiv preprint arXiv:1911.11132*, 2019. 117

James Hensman, Nicoló Fusi, and Neil D. Lawrence. Gaussian processes for big data. In Ann E. Nicholson and Padhraic Smyth (eds.), *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, UAI 2013, Bellevue, WA, USA, August 11-15, 2013*. AUAI Press, 2013. URL https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=2389&proceeding_id=29. 34

Tom Heskes. Practical confidence and prediction intervals. In Michael Mozer, Michael I. Jordan, and Thomas Petsche (eds.), *Advances in Neural Information Processing Systems 9, NIPS, Denver, CO, USA, December 2-5, 1996*, pp. 176–182. MIT Press, 1996. URL http://papers.nips.cc/paper/1306-practical-confidence-and-prediction-intervals. 31

Geoffrey Hinton and Drew van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In Lenny Pitt (ed.), *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory, COLT 1993, Santa Cruz, CA, USA, July 26-28, 1993*, pp. 5–13. ACM, 1993. doi: 10.1145/168304.168306. URL https://doi.org/10.1145/168304.168306. 33

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012. 42

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9 (8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735. 13, 42, 44

Kenneth Holstein, Jennifer Wortman Vaughan, Hal Daumé III, Miroslav Dudik, and Hanna M. Wallach. Improving fairness in machine learning systems: What do industry practitioners need? In Stephen A. Brewster, Geraldine Fitzpatrick, Anna L. Cox, and Vassilis Kostakos (eds.), *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI 2019, Glasgow, Scotland, UK, May 04-09, 2019*, pp. 600. ACM, 2019. doi: 10.1145/3290605.3300830. URL https://doi.org/10.1145/3290605.3300830. 95

Kurt Hornik, Maxwell B. Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. doi: 10.1016/0893-6080(89)90020-8. URL https://doi.org/10.1016/0893-6080(89)90020-8. 15

Sebastian Houben, Stephanie Abrecht, Maram Akila, Andreas Bär, Felix Brockherde, Patrick Feifel, Tim Fingscheidt, Sujan Sai Gannamaneni, Seyed Eghbal Ghobadi, Ahmed Hammam, et al. Inspect, understand, overcome: A survey of practical methods for AI safety. In *Deep Neural Networks and Data for Automated Driving - Robustness, Uncertainty Quantification, and Insights Towards Safety*. Springer International Publishing, 2022. 1, 95

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 2261–2269. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.243. URL https://doi.org/10.1109/CVPR.2017.243. 35

Kejun Huang, Xiao Fu, and Nicholas D. Sidiropoulos. Learning hidden markov models from pairwise co-occurrences with application to topic modeling. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2073–2082. PMLR, 2018. URL http://proceedings.mlr.press/v80/huang18c.html. 44, 48

Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics*, pp. 492–518. Springer, 1992. 17

Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Mach. Learn.*, 110(3):457–506, 2021. doi: 10.1007/s10994-021-05946-3. URL https://doi.org/10.1007/s10994-021-05946-3. 27, 28

Eddy Ilg, Özgün Çiçek, Silvio Galesso, Aaron Klein, Osama Makansi, Frank Hutter, and Thomas Brox. Uncertainty estimates and multi-hypotheses networks for optical flow. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss (eds.), *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany,*

September 8-14, 2018, Proceedings, Part VII, volume 11211 of *Lecture Notes in Computer Science*, pp. 677–693. Springer, 2018. doi: 10.1007/978-3-030-01234-2_40. URL https://doi.org/10.1007/978-3-030-01234-2_40. 109

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 448–456. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/ioffe15.html. 24

Fuyuki Ishikawa and Nobukazu Yoshioka. How do engineers perceive difficulties in engineering of machine-learning systems?: questionnaire survey. In Marcus Ciolkowski, Dusica Marijan, Matthias Galster, Weiyi Shang, Andreas Jedlitschka, Rakesh Shukla, and Kanchana Padmanabhan (eds.), *Proceedings of the Joint 7th International Workshop on Conducting Empirical Studies in Industry and 6th International Workshop on Software Engineering Research and Industrial Practice, CESSER-IP@ICSE 2019, Montreal, QC, Canada, May 27, 2019*, pp. 2–9. IEEE / ACM, 2019. doi: 10.1109/CESSER-IP.2019.00009. URL https://dl.acm.org/citation.cfm?id=3338708. 95

ISO/IEC JTC 1/SC 7. ISO/IEC 25000:2014: Systems and software engineering - systems and software quality requirements and evaluation (SQuaRE) - guide to SQuaRE, 2014. URL https://www.iso.org/standard/64764.html. 96

ISO TC 22/SC 32. ISO/AWI PAS 8800: Road vehicles — safety and artificial intelligence, 2021. URL https://www.iso.org/standard/83303.html. 95

ISO/IEC JTC 1/SC 42. Standardization in the area of artificial intelligence, 2017. URL https://www.iso.org/committee/6794475.html. 92, 94

ISO/IEC JTC 1/SC 42. ISO/IEC TR 24028:2020: Information technology - artificial intelligence - overview of trustworthiness in artificial intelligence, 2020. URL https://www.iso.org/standard/77608.html. 94

ISO/IEC JTC 1/SC 42. ISO/IEC TR 24029-1:2021: Artificial intelligence - assessment of the robustness of neural networks - part 1: Overview, 2021a. URL https://www.iso.org/standard/77609.html. 94

ISO/IEC JTC 1/SC 42. ISO/IEC CD 42001: Information technology - artificial intelligence - management system, 2021b. URL https://www.iso.org/standard/81230.html. 94

ISO/IEC JTC 1/SC 7. ISO/IEC/IEEE 15026-1:2019: Systems and software engineering - systems and software assurance - part 1: Concepts and vocabulary, 2019. URL https://www.iso.org/standard/73567.html. 95

ISO/TC 22/SC 32. ISO 26262-1:2018: Road vehicles - functional safety - part 1: Vocabulary, 2018. URL https://www.iso.org/standard/68383.html. 96

Jessica S Jermakian and David S Zuby. Primary pedestrian crash scenarios: factors relevant to the design of pedestrian detection systems. *Insurance Institute for Highway Safety, Arlington, VA*, 2011. 121

Weiqi Ji, Zhuyin Ren, and Chung K. Law. Uncertainty propagation in deep neural network using active subspace. *CoRR*, abs/1903.03989, 2019. URL `http://arxiv.org/abs/1903.03989`. 101

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Zidek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873): 583–589, 2021. 13

Michael Kamp, Linara Adilova, Joachim Sicking, Fabian Hüger, Peter Schlicht, Tim Wirtz, and Stefan Wrobel. Efficient decentralized deep learning by dynamic model averaging. In Michele Berlingerio, Francesco Bonchi, Thomas Gärtner, Neil Hurley, and Georgiana Ifrim (eds.), *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2018, Dublin, Ireland, September 10-14, 2018, Proceedings, Part I*, volume 11051 of *Lecture Notes in Computer Science*, pp. 393–409. Springer, 2018. doi: 10.1007/978-3-030-10925-7_24. URL `https://doi.org/10.1007/978-3-030-10925-7_24`. 112

Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-Sim: Learning to generate synthetic datasets. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 4550–4559. IEEE, 2019. doi: 10.1109/ICCV.2019.00465. URL `https://doi.org/10.1109/ICCV.2019.00465`. 121

Archit Karandikar, Nicholas Cain, Dustin Tran, Balaji Lakshminarayanan, Jonathon Shlens, Michael C. Mozer, and Becca Roelofs. Soft calibration objectives for neural networks. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 29768–29779, 2021. URL `https://proceedings.neurips.cc/paper/2021/hash/f8905bd3df64ace64a68e154ba72f24c-Abstract.html`. 40

R. Kavitha, V.R. Kavitha, and N. Suresh Kumar. Requirement based test case prioritization. In *2010 International Conference on Communication Control and Computing Technologies*, pp. 826–829, 2010. doi: 10.1109/ICCCCT.2010.5670728. 92

Tim Kelly and Rob Weaver. The goal structuring notation – a safety argument notation. In *Proceedings of the Dependable Systems and Networks 2004 Workshop on Assurance Cases*, pp. 6. Citeseer, 2004. 124

Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5574–5584, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/2650d6089a6d640c5e85b2b88265dc2b-Abstract.html. 3, 35, 40, 61, 72, 111

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=H1oyRlYgg. 19

Patrick Kidger and Terry J. Lyons. Universal approximation with deep narrow networks. In Jacob D. Abernethy and Shivani Agarwal (eds.), *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, volume 125 of *Proceedings of Machine Learning Research*, pp. 2306–2327. PMLR, 2020. URL http://proceedings.mlr.press/v125/kidger20a.html. 15

Jack Kiefer and Jacob Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, pp. 462–466, 1952. 18

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie J. Cai, James Wexler, Fernanda B. Viégas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2673–2682. PMLR, 2018. URL http://proceedings.mlr.press/v80/kim18d.html. 95

Byol Kim, Chen Xu, and Rina Barber. Predictive inference is free with the jackknife+-after-bootstrap. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 4138–4149. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/2b346a0aa375a07f5a90a344a61416c4-Paper.pdf. 38

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1181. URL https://www.aclweb.org/anthology/D14-1181. 42

Yoon Kim, Carl Denton, Luong Hoang, and Alexander M. Rush. Structured attention networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=HkE0Nvqlg. 42

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980. 19, 48, 55, 72

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations*, 2014. 34

Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch, and Simon Winwood. seL4: formal verification of an OS kernel. In Jeanna Neefe Matthews and Thomas E. Anderson (eds.), *Proceedings of the 22nd ACM Symposium on Operating Systems Principles 2009, SOSP 2009, Big Sky, Montana, USA, October 11-14, 2009*, pp. 207–220. ACM, 2009. doi: 10.1145/1629575.1629596. URL https://doi.org/10.1145/1629575.1629596. 92

Moritz Klischat and Matthias Althoff. Generating critical test scenarios for automated vehicles with evolutionary algorithms. In *2019 IEEE Intelligent Vehicles Symposium, IV 2019, Paris, France, June 9-12, 2019*, pp. 2352–2358. IEEE, 2019. doi: 10.1109/IVS.2019.8814230. URL https://doi.org/10.1109/IVS.2019.8814230. 121

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton Earnshaw, Imran Haque, Sara M. Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. WILDS: A benchmark of in-the-wild distribution shifts. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5637–5664. PMLR, 2021. URL http://proceedings.mlr.press/v139/koh21a.html. 131

Simon Kohl, Bernardino Romera-Paredes, Clemens Meyer, Jeffrey De Fauw, Joseph R. Ledsam, Klaus H. Maier-Hein, S. M. Ali Eslami, Danilo Jimenez Rezende, and Olaf Ronneberger. A probabilistic u-net for segmentation of ambiguous images. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 6965–6975, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/473447ac58e1cd7e96172575f48dca3b-Abstract.html. 3, 112, 113

Philip Koopman. The heavy tail safety ceiling. In *Automated and Connected Vehicle Systems Testing Symposium*, volume 1145, 2018. 1

Philip Koopman and Frank Fratrik. How many operational design domains, objects, and events? In Huáscar Espinoza, Seán Ó hÉigeartaigh, Xiaowei Huang, José Hernández-

Orallo, and Mauricio Castillo-Effen (eds.), *Workshop on Artificial Intelligence Safety 2019 co-located with the Thirty-Third AAAI Conference on Artificial Intelligence 2019 (AAAI-19), Honolulu, Hawaii, January 27, 2019*, volume 2301 of *CEUR Workshop Proceedings*, 2019. URL http://ceur-ws.org/Vol-2301/paper_6.pdf. 99

Philip Koopman and Beth Osyk. Safety argument considerations for public road testing of autonomous vehicles. *SAE Technical Paper Series*, 2019. 95

Philip Koopman, Uma Ferrell, Frank Fratrik, and Michael D. Wagner. A safety standard approach for fully autonomous vehicles. In Alexander B. Romanovsky, Elena Troubitsyna, Ilir Gashi, Erwin Schoitsch, and Friedemann Bitsch (eds.), *Computer Safety, Reliability, and Security - SAFECOMP 2019 Workshops, ASSURE, DECSoS, SASSUR, STRIVE, and WAISE, Turku, Finland, September 10, 2019, Proceedings*, volume 11699 of *Lecture Notes in Computer Science*, pp. 326–332. Springer, 2019. doi: 10.1007/978-3-030-26250-1_26. URL https://doi.org/10.1007/978-3-030-26250-1_26. 95

Ranganath Krishnan and Omesh Tickoo. Improving model calibration with accuracy versus uncertainty optimization. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/d3d9446802a44259755d38e6d163e820-Abstract.html. 40

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pp. 1106–1114, 2012. URL https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html. 17

Nathan P. Kropp, Philip J. Koopman Jr., and Daniel P. Siewiorek. Automated robustness testing of off-the-shelf software components. In *Digest of Papers: FTCS-28, The Twenty-Eigth Annual International Symposium on Fault-Tolerant Computing, Munich, Germany, June 23-25, 1998*, pp. 230–239. IEEE Computer Society, 1998. doi: 10.1109/FTCS.1998.689474. URL https://doi.org/10.1109/FTCS.1998.689474. 92

D. Richard Kuhn, Dolores R. Wallace, and Albert M. Gallo. Software fault interactions and implications for software testing. *IEEE Trans. Software Eng.*, 30(6):418–421, 2004. doi: 10.1109/TSE.2004.24. URL https://doi.org/10.1109/TSE.2004.24. 120

Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80

of *Proceedings of Machine Learning Research*, pp. 2796–2804. PMLR, 7 2018. URL http://proceedings.mlr.press/v80/kuleshov18a.html. 32, 39

Ananya Kumar, Percy Liang, and Tengyu Ma. Verified uncertainty calibration. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 3787–3798, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/f8c0c968632845cd133308b1a494967f-Abstract.html. 32

Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. Trainable calibration measures for neural networks from kernel mean embeddings. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2810–2819. PMLR, 2018. URL http://proceedings.mlr.press/v80/kumar18a.html. 40

Balaji Lakshminarayanan and Raviv Raich. Non-negative matrix factorization for parameter estimation in hidden Markov models. In *2010 IEEE International Workshop on Machine Learning for Signal Processing*, pp. 89–94. IEEE, 2010. 44

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6402–6413, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/9ef2ed4b7fd2c810847ffa5fa85bce38-Abstract.html. 37, 61, 72, 111

Henry Jacob Landau. Sampling, data transmission, and the Nyquist rate. *Proceedings of the IEEE*, 55(10):1701–1706, 1967. doi: 10.1109/PROC.1967.5962. 66

Quoc V. Le and Tomás Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pp. 1188–1196. JMLR.org, 2014. URL http://proceedings.mlr.press/v32/le14.html. 42

Colin Lea, René Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks: A unified approach to action segmentation. In Gang Hua and Hervé Jégou (eds.), *Computer Vision - ECCV 2016 Workshops - Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III*, volume 9915 of *Lecture Notes in Computer Science*, pp. 47–54, 2016. doi: 10.1007/978-3-319-49409-8_7. URL https://doi.org/10.1007/978-3-319-49409-8_7. 17

Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Handwritten digit recognition with a back-propagation network. In David S. Touretzky (ed.), *Advances in Neural Information Processing Systems 2, [NIPS Conference, Denver, Colorado, USA, November 27-30, 1989]*, pp. 396–404, 1989. URL `http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network`. 17

Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In David A. Forsyth, Joseph L. Mundy, Vito Di Gesù, and Roberto Cipolla (eds.), *Shape, Contour and Grouping in Computer Vision*, volume 1681 of *Lecture Notes in Computer Science*, pp. 319. Springer, 1999. doi: 10.1007/3-540-46805-6_19. URL `https://doi.org/10.1007/3-540-46805-6_19`. 15

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015. 42

Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999. 44

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as Gaussian processes. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL `https://openreview.net/forum?id=B1EA-M-0Z`. 54

Jason D. Lee, Max Simchowitz, Michael I. Jordan, and Benjamin Recht. Gradient descent only converges to minimizers. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir (eds.), *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, volume 49 of *JMLR Workshop and Conference Proceedings*, pp. 1246–1257. JMLR.org, 2016. URL `http://proceedings.mlr.press/v49/lee16.html`. 19

Brian G Leroux. Maximum-likelihood estimation for hidden Markov models. *Stochastic Processes and Their Applications*, 40(1):127–143, 1992. 42

David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In W. Bruce Croft and C. J. van Rijsbergen (eds.), *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum)*, pp. 3–12. ACM/Springer, 1994. doi: 10.1007/978-1-4471-2099-5_1. URL `https://doi.org/10.1007/978-1-4471-2099-5_1`. 92

Yang Li and Tao Yang. Word embedding for understanding natural language: a survey. In *Guide to Big Data Applications*, pp. 83–104. Springer, 2018. 42

Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 936–944. IEEE Computer Society, 2017a. doi: 10.1109/CVPR.2017.106. URL https://doi.org/10.1109/CVPR.2017.106. 87

Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 2999–3007. IEEE Computer Society, 2017b. doi: 10.1109/ICCV.2017.324. URL https://doi.org/10.1109/ICCV.2017.324. 7, 17, 18, 82, 87

Jeremiah Z. Liu, John W. Paisley, Marianthi-Anna Kioumourtzoglou, and Brent A. Coull. Accurate uncertainty estimation and decomposition in ensemble learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 8950–8961, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/1cc8a8ea51cd0adddf5dab504a285915-Abstract.html. 27, 38, 112

Li Liu, Wanli Ouyang, Xiaogang Wang, Paul W. Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *Int. J. Comput. Vis.*, 128(2):261–318, 2020. doi: 10.1007/s11263-019-01247-4. URL https://doi.org/10.1007/s11263-019-01247-4. 82

Siqi Liu, Sidong Liu, Weidong Cai, Sonia Pujol, Ron Kikinis, and Dagan Feng. Early diagnosis of Alzheimer's disease with deep learning. In *IEEE 11th International Symposium on Biomedical Imaging, ISBI 2014, April 29 - May 2, 2014, Beijing, Chin, Beijing, China*, pp. 1015–1018. IEEE, 2014. doi: 10.1109/ISBI.2014.6868045. URL https://doi.org/10.1109/ISBI.2014.6868045. 1

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (eds.), *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I*, volume 9905 of *Lecture Notes in Computer Science*, pp. 21–37. Springer, 2016. doi: 10.1007/978-3-319-46448-0_2. URL https://doi.org/10.1007/978-3-319-46448-0_2. 17, 18

Charles F. Van Loan. The ubiquitous Kronecker product. *Journal of Computational and Applied Mathematics*, 123(1):85–100, 2000. ISSN 0377-0427. doi: https://doi.org/10.1016/S0377-0427(00)00393-9. URL https://www.sciencedirect.com/science/article/pii/S0377042700003939. Numerical Analysis 2000. Vol. III: Linear Algebra. 33

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 3431–3440. IEEE Computer Society, 2015. doi: 10.1109/CVPR.2015.7298965. URL https://doi.org/10.1109/CVPR.2015.7298965. 17

Ping Luo, Xinjiang Wang, Wenqi Shao, and Zhanglin Peng. Towards understanding regularization in batch normalization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=HJlLKjR9FQ. 24

Lucy Ellen Lwakatare, Aiswarya Raj, Ivica Crnkovic, Jan Bosch, and Helena Holmström Olsson. Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions. *Inf. Softw. Technol.*, 127:106368, 2020. doi: 10.1016/j.infsof.2020.106368. URL https://doi.org/10.1016/j.infsof.2020.106368. 95

Yi-An Ma, Nicholas J. Foti, and Emily B. Fox. Stochastic gradient MCMC methods for hidden Markov models. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2265–2274. PMLR, 2017. URL http://proceedings.mlr.press/v70/ma17a.html. 44

Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, Atlanta, Georgia, 2013. 15

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=S1jE5L5gl. 35

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for Bayesian uncertainty in deep learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/file/118921efba23fc329e6560b27861f0c2-Paper.pdf. 4, 37, 72, 113

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rJzIBfZAb. 95, 125

Mehrdad Majzoobi, Farinaz Koushanfar, and Miodrag Potkonjak. Testing techniques for hardware security. In Douglas Young and Nur A. Touba (eds.), *2008 IEEE International Test Conference, ITC 2008, Santa Clara, California, USA, October 26-31, 2008*, pp. 1–10. IEEE Computer Society, 2008. doi: 10.1109/TEST.2008.4700636. URL https://doi.org/10.1109/TEST.2008.4700636. 92

Andrey Malinin and Mark J. F. Gales. Predictive uncertainty estimation via prior networks. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 7047–7058, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/3ea2db50e62ceefceaf70a9d9a56a6f4-Abstract.html. 27, 90, 111, 112

Andrey Malinin, Bruno Mlodozeniec, and Mark J. F. Gales. Ensemble distribution distillation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=BygSP6Vtvr. 4

Andrey Malinin, Neil Band, Yarin Gal, Mark J. F. Gales, Alexander Ganshin, German Chesnokov, Alexey Noskov, Andrey Ploskonosov, Liudmila Prokhorenkova, Ivan Provilkov, Vatsal Raina, Vyas Raina, Denis Roginskiy, Mariya Shmatova, Panagiotis Tigas, and Boris Yangel. Shifts: A dataset of real distributional shift across multiple large-scale tasks. In Joaquin Vanschoren and Sai-Kit Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/ad61ab143223efbc24c7d2583be69251-Abstract-round2.html. 131

Stephan Mandt, Matthew D. Hoffman, and David M. Blei. Stochastic gradient descent as approximate Bayesian inference. *J. Mach. Learn. Res.*, 18:134:1–134:35, 2017. URL http://jmlr.org/papers/v18/17-214.html. 36

James Martens and Roger B. Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In Francis R. Bach and David M. Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 2408–2417. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/martens15.html. 33

Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *CoRR*, abs/1804.07612, 2018. URL http://arxiv.org/abs/1804.07612. 19

Alexander Matthews, Jiri Hron, Mark Rowland, Richard E. Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada,*

April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net, 2018. URL https://openreview.net/forum?id=H1-nGgWC-. 54, 55

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In Aarti Singh and Xiaojin (Jerry) Zhu (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282. PMLR, 2017. URL http://proceedings.mlr.press/v54/mcmahan17a.html. 112

Hendrik Meth, Benjamin Müller, and Alexander Maedche. Designing a requirement mining system. *J. Assoc. Inf. Syst.*, 16(9):2, 2015. URL http://aisel.aisnet.org/jais/vol16/iss9/2. 92

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953. 36

Frederik Meudt, Martin Theissen, Rudi Schäfer, and Thomas Guhr. Constructing analytically tractable ensembles of stochastic covariances with an application to financial data. *Journal of Statistical Mechanics: Theory and Experiment*, 2015(11): P11025, nov 2015. doi: 10.1088/1742-5468/2015/11/p11025. URL https://doi.org/10.1088/1742-5468/2015/11/p11025. 131

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun (eds.), *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013a. URL http://arxiv.org/abs/1301.3781. 42

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pp. 3111–3119, 2013b. URL https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html. 6, 42, 45, 127

Tomás Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association For Computational Linguistics: Human Language Technologies*, pp. 746–751, 2013c. 42

Seonwoo Min, Seunghyun Park, Siwon Kim, Hyun-Soo Choi, and Sungroh Yoon. Pre-training of deep bidirectional protein sequence representations with structural information. *Neural Information Processing Systems, Workshop on Learning Meaningful Representations of Life*, 2019. 53

Matiur Rahman Minar and Jibon Naher. Recent advances in deep learning: An overview. *arXiv preprint arXiv:1807.08169*, 2018. 42

Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In Danah Boyd and Jamie H. Morgenstern (eds.), *Proceedings of the Conference on Fairness, Accountability, and Transparency, FAT* 2019, Atlanta, GA, USA, January 29-31, 2019*, pp. 220–229. ACM, 2019. doi: 10.1145/3287560.3287596. URL `https://doi.org/10.1145/3287560.3287596`. 95, 130

Michael Mock, Stephan Scholz, Frédérik Blank, Fabian Hüger, Andreas J. Rohatschek, Loren Schwarz, and Thomas Stauner. An integrated approach to a safety argumentation for AI-based perception functions in automated driving. In Ibrahim Habli, Mark Sujan, Simos Gerasimou, Erwin Schoitsch, and Friedemann Bitsch (eds.), *Computer Safety, Reliability, and Security. SAFECOMP 2021 Workshops - DECSoS, MAPSOD, DepDevOps, USDAI, and WAISE, York, UK, September 7, 2021, Proceedings*, volume 12853 of *Lecture Notes in Computer Science*, pp. 265–271. Springer, 2021. doi: 10.1007/978-3-030-83906-2_21. URL `https://doi.org/10.1007/978-3-030-83906-2_21`. 95

Miguel Monteiro, Loïc Le Folgoc, Daniel Coelho de Castro, Nick Pawlowski, Bernardo Marques, Konstantinos Kamnitsas, Mark van der Wilk, and Ben Glocker. Stochastic segmentation networks: Modelling spatially correlated aleatoric uncertainty. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/95f8d9901ca8878e291552f001f67692-Abstract.html`. 113

Seokyong Moon and Jenq-Neng Hwang. Robust speech recognition based on joint model and feature space optimization of hidden Markov models. *IEEE Transactions on Neural Networks*, 8(2):194–204, 1997. 44

Nelson Morgan and Herve Bourlard. Generalization and parameter estimation in feedforward nets: Some experiments. In D. Touretzky (ed.), *Advances in Neural Information Processing Systems*, volume 2, 1990. URL `https://proceedings.neurips.cc/paper/1989/file/63923f49e5241343aa7acb6a06a751e7-Paper.pdf`. 24

Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using Bayesian binning. In Blai Bonet and Sven Koenig (eds.),

Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA, pp. 2901–2907. AAAI Press, 2015. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9667. 39, 105

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=B1g5sA4twr. 23

Radford M. Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 54:113–162, 2010. 36

Clémentine Nebut, Franck Fleurey, Yves Le Traon, and Jean-Marc Jézéquel. A requirement-based approach to test product families. In Frank van der Linden (ed.), *Software Product-Family Engineering, 5th International Workshop, PFE 2003, Siena, Italy, November 4-6, 2003, Revised Papers*, volume 3014 of *Lecture Notes in Computer Science*, pp. 198–210. Springer, 2003. doi: 10.1007/978-3-540-24667-1_15. URL https://doi.org/10.1007/978-3-540-24667-1_15. 92

Yurii E. Nesterov. *Introductory Lectures on Convex Optimization - A Basic Course*, volume 87 of *Applied Optimization*. Springer, 2004. ISBN 978-1-4613-4691-3. doi: 10.1007/978-1-4419-8853-9. URL https://doi.org/10.1007/978-1-4419-8853-9. 19

Lukás Neumann, Michelle Karg, Shanshan Zhang, Christian Scharfenberger, Eric Piegert, Sarah Mistr, Olga Prokofyeva, Robert Thiel, Andrea Vedaldi, Andrew Zisserman, and Bernt Schiele. Nightowls: A pedestrians at night dataset. In C. V. Jawahar, Hongdong Li, Greg Mori, and Konrad Schindler (eds.), *Computer Vision - ACCV 2018 - 14th Asian Conference on Computer Vision, Perth, Australia, December 2-6, 2018, Revised Selected Papers, Part I*, volume 11361 of *Lecture Notes in Computer Science*, pp. 691–705. Springer, 2018. doi: 10.1007/978-3-030-20887-5_43. URL https://doi.org/10.1007/978-3-030-20887-5_43. 84

U.S. National Highway Traffic Safety Administration NHTSA. Automated driving systems 2.0: A vision for safety, 2017. URL https://www.nhtsa.gov/sites/nhtsa.gov/files/documents/13069a-ads2.0_090617_v9a_tag.pdf. (accessed on 2022-01-14). 99

David A. Nix and Andreas S. Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 1, pp. 55–60 vol.1, 1994. doi: 10.1109/ICNN.1994.374138. 31, 72, 112

Curtis G. Northcutt, Lu Jiang, and Isaac L. Chuang. Confident learning: Estimating uncertainty in dataset labels. *J. Artif. Intell. Res.*, 70:1373–1411, 2021. doi: 10.1613/jair.1.12125. URL https://doi.org/10.1613/jair.1.12125. 131

U.S. National Transportation Safety Board NTSB. Highway accident brief - collision between car operating with partial driving automation and truck-tractor semitrailer, Delray Beach, Florida, March 1, 2019, 2019a. URL `https://www.ntsb.gov/investigations/AccidentReports/Reports/HAB2001.pdf`. 1

U.S. National Transportation Safety Board NTSB. Accident report - collision between vehicle controlled by developmental automated driving system and pedestrian, Tempe, Arizona March 18, 2018, 2019b. URL `https://www.ntsb.gov/investigations/accidentreports/reports/har1903.pdf`. 1

Rizki Nurfauzi, Hanung Adi Nugroho, Igi Ardiyanto, and Eka Legya Frannita. Autocorrection of lung boundary on 3D CT lung cancer images. *J. King Saud Univ. Comput. Inf. Sci.*, 33(5):518–527, 2021. doi: 10.1016/j.jksuci.2019.02.009. URL `https://doi.org/10.1016/j.jksuci.2019.02.009`. 3

Augustus Odena, Catherine Olsson, David G. Andersen, and Ian J. Goodfellow. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4901–4911. PMLR, 2019. URL `http://proceedings.mlr.press/v97/odena19a.html`. 96, 121

Manfred Opper and Cédric Archambeau. The variational Gaussian approximation revisited. *Neural Comput.*, 21(3):786–792, 2009. doi: 10.1162/neco.2008.08-07-592. URL `https://doi.org/10.1162/neco.2008.08-07-592`. 34

Ian Osband. Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout. In *Neural Information Processing Systems Workshop on Bayesian Deep Learning*, volume 192, 2016. 6

Dhirendra Pandey, U. Suman, and A.K. Ramani. An effective requirement engineering process model for software development and requirements management. In *2010 International Conference on Advances in Recent Technologies in Communication and Computing*, pp. 287–291, 2010. doi: 10.1109/ARTCom.2010.24. 92

Moschos Papananias, Thomas E. McLeay, Mahdi Mahfouf, and Visakan Kadirkamanathan. A Bayesian framework to estimate part quality and associated uncertainties in multistage manufacturing. *Comput. Ind.*, 105:35–47, 2019. doi: 10.1016/j.compind.2018.10.008. URL `https://doi.org/10.1016/j.compind.2018.10.008`. 91

Jungin Park, Jiyoung Lee, Ig-Jae Kim, and Kwanghoon Sohn. Probabilistic representations for video contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14711–14721, 2022. 132

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison,

Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 8024–8035, 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html`. 35, 141

Sandeep Paul, Lotika Singh, et al. A review on advances in deep learning. In *2015 IEEE Workshop on Computational Intelligence: Theories, Applications and Future Directions (WCI)*, pp. 1–6. IEEE, 2015. 42

Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pp. 1–18. ACM, 2017. doi: 10.1145/3132747.3132785. URL `https://doi.org/10.1145/3132747.3132785`. 96, 121

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pp. 1532–1543. ACL, 2014. doi: 10.3115/v1/d14-1162. URL `https://doi.org/10.3115/v1/d14-1162`. 42

Thomas Perneger. The Swiss cheese model of safety incidents: Are there holes in the metaphor? *BMC Health Services Research*, 5:71, 02 2005. doi: 10.1186/1472-6963-5-71. 104

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. DeepWalk: online learning of social representations. In Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani (eds.), *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pp. 701–710. ACM, 2014. doi: 10.1145/2623330.2623732. URL `https://doi.org/10.1145/2623330.2623732`. 42

Maximilian Pintz. Novel optimization schemes for uncertainty estimation in regression neural networks. *Master's Thesis, Computer Science, University of Bonn*, 2021. 4

Janis Postels, Francesco Ferroni, Huseyin Coskun, Nassir Navab, and Federico Tombari. Sampling-free epistemic uncertainty estimation using approximated variance propagation. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 2931–2940. IEEE, 2019. doi:

10.1109/ICCV.2019.00302. URL https://doi.org/10.1109/ICCV.2019.00302. 36, 90, 112

Janis Postels, Mattia Segù, Tao Sun, Luca Daniel Sieber, Luc Van Gool, Fisher Yu, and Federico Tombari. On the practicality of deterministic epistemic uncertainty. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 17870–17909. PMLR, 2022. URL https://proceedings.mlr.press/v162/postels22a.html. 32

Maoying Qiao, Wei Bian, Richard Yi Da Xu, and Dacheng Tao. Diversified hidden Markov models for sequential labeling. *IEEE Trans. Knowl. Data Eng.*, 27(11):2947–2960, 2015. doi: 10.1109/TKDE.2015.2433262. URL https://doi.org/10.1109/TKDE.2015.2433262. 44

Maurice Henri Quenouille. Notes on bias in estimation. *Biometrika*, 43(3-4):353–360, 12 1956. ISSN 0006-3444. doi: 10.1093/biomet/43.3-4.353. URL https://doi.org/10.1093/biomet/43.3-4.353. 38

Lawrence R Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. 44, 50

Lawrence R Rabiner and Biing-Hwang Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986. 42, 44, 127

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, H. Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant M. Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake A. Hechtman, Laura Weidinger, Iason Gabriel, William S. Isaac, Edward Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. Scaling language models: Methods, analysis & insights from training gopher. *CoRR*, abs/2112.11446, 2021. URL https://arxiv.org/abs/2112.11446. 132

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *CoRR*, abs/2204.06125, 2022. doi: 10.48550/arXiv.2204.06125. URL https://doi.org/10.48550/arXiv.2204.06125. 132

Carl Edward Rasmussen. Gaussian processes in machine learning. In Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch (eds.), *Advanced Lectures on Machine Learning, ML Summer Schools 2003, Canberra, Australia, February 2-14, 2003, Tübingen, Germany, August 4-16, 2003, Revised Lectures*, volume 3176 of *Lecture Notes in Computer Science*, pp. 63–71. Springer, 2003. doi: 10.1007/978-3-540-28650-9_4. URL https://doi.org/10.1007/978-3-540-28650-9_4. 34

Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 779–788. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.91. URL https://doi.org/10.1109/CVPR.2016.91. 82

Scott E. Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *CoRR*, abs/2205.06175, 2022. doi: 10.48550/arXiv.2205.06175. URL https://doi.org/10.48550/arXiv.2205.06175. 132

Philipp Reinkemeier, Ingo Stierand, Philip Rehkop, and Stefan Henkler. A pattern-based requirement specification language: Mapping automotive specific timing requirements. In Ralf H. Reussner, Alexander Pretschner, and Stefan Jähnichen (eds.), *Software Engineering 2011 - Workshopband (inkl. Doktorandensymposium), Fachtagung des GI-Fachbereichs Softwaretechnik, 21.-25.02.2011, Karlsruhe*, volume P-184 of *LNI*, pp. 99–108. GI, 2011. URL https://dl.gi.de/20.500.12116/19877. 104

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 91–99, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/14bfa6bb14875e45bba028a21ed38046-Abstract.html. 82

Vincenzo Riccio, Gunel Jahangirova, Andrea Stocco, Nargiz Humbatova, Michael Weiss, and Paolo Tonella. Testing machine learning based systems: a systematic mapping. *Empir. Softw. Eng.*, 25(6):5193–5254, 2020. doi: 10.1007/s10664-020-09881-0. URL https://doi.org/10.1007/s10664-020-09881-0. 95

Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable Laplace approximation for neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=Skdvd2xAZ. 33

Daniel A. Roberts, Sho Yaida, and Boris Hanin. The principles of deep learning theory. *CoRR*, abs/2106.10165, 2021. URL https://arxiv.org/abs/2106.10165. 15

R. Tyrrell Rockafellar and Stanislav Uryasev. Conditional value-at-risk for general loss distributions. *Journal of Banking & Finance*, 26(7):1443–1471, 2002. ISSN 0378-4266. doi: https://doi.org/10.1016/S0378-4266(02)00271-6. URL https://www.sciencedirect.com/science/article/pii/S0378426602002716. 71

Rebecca Roelofs, Nicholas Cain, Jonathon Shlens, and Michael C. Mozer. Mitigating bias in calibration error estimation. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera (eds.), *International Conference on Artificial Intelligence and Statistics, AISTATS 2022, 28-30 March 2022, Virtual Event*, volume 151 of *Proceedings of Machine Learning Research*, pp. 4036–4054. PMLR, 2022. URL https://proceedings.mlr.press/v151/roelofs22a.html. 39

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958. 13, 16

Arpan Roy, Dong Seong Kim, and Kishor S. Trivedi. Cyber security analysis using attack countermeasure trees. In Frederick T. Sheldon, Stacy J. Prowell, Robert K. Abercrombie, and Axel W. Krings (eds.), *Proceedings of the 6th Cyber Security and Information Intelligence Research Workshop, CSIIRW 2010, Oak Ridge, TN, USA, April 21-23, 2010*, pp. 28. ACM, 2010. doi: 10.1145/1852666.1852698. URL https://doi.org/10.1145/1852666.1852698. 126

Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. A metric for distributions with applications to image databases. In *Proceedings of the Sixth International Conference on Computer Vision (ICCV-98), Bombay, India, January 4-7, 1998*, pp. 59–66. IEEE Computer Society, 1998. doi: 10.1109/ICCV.1998.710701. URL https://doi.org/10.1109/ICCV.1998.710701. 69

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985. 13

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986. 18

Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. A convex relaxation barrier to tight robustness verification of neural networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox,

and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 9832–9842, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/246a3c5544feb054f3ea718f61adfa16-Abstract.html. 106

Jerome H Saltzer and Michael D Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975. 43

John M Scanlon, Kristofer D Kusano, Tom Daniel, Christopher Alderson, Alexander Ogle, and Trent Victor. Waymo simulated driving behavior in reconstructed fatal crashes within an autonomous vehicle operating domain, 2021. 121

Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden Markov models for information extraction. In Frank Hoffmann, David J. Hand, Niall M. Adams, Douglas H. Fisher, and Gabriela Guimarães (eds.), *Advances in Intelligent Data Analysis, 4th International Conference, IDA 2001, Cascais, Portugal, September 13-15, 2001, Proceedings*, volume 2189 of *Lecture Notes in Computer Science*, pp. 309–318. Springer, 2001. doi: 10.1007/3-540-44816-0_31. URL https://doi.org/10.1007/3-540-44816-0_31. 42

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. doi: 10.1016/j.neunet.2014.09.003. URL https://doi.org/10.1016/j.neunet.2014.09.003. 16, 42

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 2503–2511, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/86df7dcfd896fcaf2674f757a2463eba-Abstract.html. 95, 101, 114

Marwin H. S. Segler, Mike Preuss, and Mark P. Waller. Planning chemical syntheses with deep neural networks and symbolic AI. *Nat.*, 555(7698):604–610, 2018. doi: 10.1038/nature25978. URL https://doi.org/10.1038/nature25978. 13

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pp. 618–626. IEEE Computer Society, 2017. doi: 10.1109/ICCV.2017.74. URL https://doi.org/10.1109/ICCV.2017.74. 95

Sina Shafaei, Stefan Kugele, Mohd Hafeez Osman, and Alois C. Knoll. Uncertainty in machine learning: A safety perspective on autonomous driving. In Barbara Gallina, Amund Skavhaug, Erwin Schoitsch, and Friedemann Bitsch (eds.), *Computer Safety, Reliability, and Security - SAFECOMP 2018 Workshops, ASSURE, DECSoS, SASSUR, STRIVE, and WAISE, Västerås, Sweden, September 18, 2018, Proceedings*, volume 11094 of *Lecture Notes in Computer Science*, pp. 458–464. Springer, 2018. doi: 10.1007/978-3-319-99229-7_39. URL https://doi.org/10.1007/978-3-319-99229-7_39. 91

Vatsal Sharan, Sham M. Kakade, Percy Liang, and Gregory Valiant. Prediction with a short memory. In Ilias Diakonikolas, David Kempe, and Monika Henzinger (eds.), *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pp. 1074–1087. ACM, 2018. doi: 10.1145/3188745.3188954. URL https://doi.org/10.1145/3188745.3188954. 42

Beate Sick, Torsten Hothorn, and Oliver Dürr. Deep transformation models: Tackling complex regression problems with neural network based transformation models. In *25th International Conference on Pattern Recognition, ICPR 2020, Virtual Event / Milan, Italy, January 10-15, 2021*, pp. 2476–2481. IEEE, 2020. doi: 10.1109/ICPR48806.2021.9413177. URL https://doi.org/10.1109/ICPR48806.2021.9413177. 131

Joachim Sicking, Maram Akila, Tim Wirtz, Sebastian Houben, and Asja Fischer. Characteristics of Monte Carlo dropout in wide neural networks. *ICML 2020 Workshop on Uncertainty and Robustness in Deep Learning*, 2020. 55

R.L. Sielken and H.O. Hartley. Two linear programming algorithms for unbiased estimation of linear models. *Journal of the American Statistical Association*, 68(343):639–641, 1973. 17

I. Robert Sipos. Parallel stratified MCMC sampling of AR-HMMs for stochastic time series prediction. In *Proceedings of the 4th Stochastic Modeling Techniques and Data Analysis International Conference with Demographics Workshop (SMTDA 2016). Valletta, Malta: University of Malta*, pp. 361–364, 2016. 44

Lawrence H. Smith, Thomas C. Rindflesch, and W. John Wilbur. MedPost: a part-of-speech tagger for biomedical text. *Bioinform.*, 20(14):2320–2321, 2004. doi: 10.1093/bioinformatics/bth227. URL https://doi.org/10.1093/bioinformatics/bth227. 50, 52

Jasper Snoek, Yaniv Ovadia, Emily Fertig, Balaji Lakshminarayanan, Sebastian Nowozin, D. Sculley, Joshua V. Dillon, Jie Ren, and Zachary Nado. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019,*

December 8-14, 2019, Vancouver, BC, Canada, pp. 13969–13980, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/8558cb408c1d76621371888657d2eb1d-Abstract.html. 4, 32, 36, 38, 72, 111, 112

Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. *A Talk at the Stanford Artificial Project*, pp. 271–272, 1968. 16

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html. 24, 25, 34

Michael A Stephens. EDF statistics for goodness of fit and some comparisons. *Journal of the American Statistical Association*, 69(347):730–737, 1974. 81, 169

Niko Sünderhauf, Oliver Brock, Walter J. Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, and Peter Corke. The limits and potentials of deep learning for robotics. *Int. J. Robotics Res.*, 37 (4-5):405–420, 2018. doi: 10.1177/0278364918770733. URL https://doi.org/10.1177/0278364918770733. 91

Luay Ho Tahat, Atef Bader, Boris Vaysburg, and Bogdan Korel. Requirement-based automated black-box test generation. In *25th International Computer Software and Applications Conference (COMPSAC 2001), Invigorating Software Development, 8-12 October 2001, Chicago, IL, USA*, pp. 489–495. IEEE Computer Society, 2001. doi: 10.1109/CMPSAC.2001.960658. URL https://doi.org/10.1109/CMPSAC.2001.960658. 92

Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothee Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural Networks*, 111: 47–63, 2019. 42

John R. Taylor. *An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements*. University Science Books, 2 sub edition, 1996. ISBN 093570275X. 36

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. Intelligent robotics and autonomous agents. MIT Press, 2005. ISBN 978-0-262-20162-9. 91

Michalis K. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In David A. Van Dyk and Max Welling (eds.), *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009*, volume 5 of *JMLR Proceedings*, pp. 567–574. JMLR.org, 2009. URL http://proceedings.mlr.press/v5/titsias09a.html. 34

Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. In *IEEE Conference on*

Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015, pp. 648–656. IEEE Computer Society, 2015. doi: 10.1109/CVPR.2015.7298664. URL https://doi.org/10.1109/CVPR.2015.7298664. 25

Ehsan Toreini, Mhairi Aitken, Kovila P. L. Coopamootoo, Karen Elliott, Carlos Gonzalez Zelaya, and Aad van Moorsel. The relationship between trust in AI and trustworthy machine learning technologies. In Mireille Hildebrandt, Carlos Castillo, L. Elisa Celis, Salvatore Ruggieri, Linnet Taylor, and Gabriela Zanfir-Fortuna (eds.), *FAT* '20: Conference on Fairness, Accountability, and Transparency, Barcelona, Spain, January 27-30, 2020*, pp. 272–283. ACM, 2020. doi: 10.1145/3351095.3372834. URL https://doi.org/10.1145/3351095.3372834. 43

Dustin Tran, Jeremiah Liu, Michael W. Dusenberry, Du Phan, Mark Collier, Jie Ren, Kehang Han, Zi Wang, Zelda Mariet, Huiyi Hu, Neil Band, Tim G. J. Rudner, Karan Singhal, Zachary Nado, Joost van Amersfoort, Andreas Kirsch, Rodolphe Jenatton, Nithum Thain, Honglin Yuan, Kelly Buchanan, Kevin Murphy, D. Sculley, Yarin Gal, Zoubin Ghahramani, Jasper Snoek, and Balaji Lakshminarayanan. Plex: Towards reliability using pretrained large model extensions, 2022. URL https://arxiv.org/abs/2207.07411. 131

Ke M. Tran, Yonatan Bisk, Ashish Vaswani, Daniel Marcu, and Kevin Knight. Unsupervised neural hidden Markov models. In *SPNLP@EMNLP*, pp. 63–71, 2016. URL https://doi.org/10.18653/v1/W16-5907. 44

Edmondo Trentin and Marco Gori. Combining neural networks and hidden Markov models for speech recognition. *Neural Nets WIRN VIETRI-98*, pp. 63–79, 1999. 44

Michael Truong-Le, Frederik Diehl, Thomas Brunner, and Alois C. Knoll. Uncertainty estimation for deep neural object detectors in safety-critical applications. In Wei-Bin Zhang, Alexandre M. Bayen, Javier J. Sánchez Medina, and Matthew J. Barth (eds.), *21st International Conference on Intelligent Transportation Systems, ITSC 2018, Maui, HI, USA, November 4-7, 2018*, pp. 3873–3878. IEEE, 2018. doi: 10.1109/ITSC.2018.8569637. URL https://doi.org/10.1109/ITSC.2018.8569637. 88, 91

Russell Tsuchida, Fred Roosta, and Marcus Gallagher. Richer priors for infinitely wide multi-layer perceptrons. *arXiv preprint arXiv:1911.12927*, 2019. 54

John Wilder Tukey. Bias and confidence in not quite large samples. *The Annals of Mathematical Statistics*, 29(2):614 – 623, 1958. doi: 10.1214/aoms/1177706647. URL https://doi.org/10.1214/aoms/1177706647. 38

Underwriters Laboratories. Standard for evaluation of autonomous products, 2020. URL https://www.shopulstandards.com/ProductDetail.aspx?productid=UL4600. (accessed on 2022-01-14). 95

Joost van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty estimation using a single deep deterministic neural network. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9690–9700. PMLR, 2020. URL http://proceedings.mlr.press/v119/van-amersfoort20a.html. 32

Vladimir Vapnik. Principles of risk minimization for learning theory. In John E. Moody, Stephen Jose Hanson, and Richard Lippmann (eds.), *Advances in Neural Information Processing Systems 4, [NIPS Conference, Denver, Colorado, USA, December 2-5, 1991]*, pp. 831–838, 1991. URL http://papers.nips.cc/paper/506-principles-of-risk-minimization-for-learning-theory. 12

Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Statistics for Engineering and Information Science. Springer, 2000. ISBN 978-1-4419-3160-3. doi: 10.1007/978-1-4757-3264-1. URL https://doi.org/10.1007/978-1-4757-3264-1. 1

Vladimir Vapnik and Alexei Chervonenkis. *Theory of Pattern Recognition [in Russian]*. Nauka, Moscow, 1974. (German Translation: W. Wapnik & A. Tscherwonenkis, *Theorie der Zeichenerkennung*, Akademie–Verlag, Berlin, 1979). 24

Kush R Varshney. Trustworthy machine learning and artificial intelligence. *XRDS: Crossroads, The ACM Magazine for Students*, 25(3):26–29, 2019. 43

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html. 13, 42

Petar Velickovic. Pgf/tikz figures, 2016. https://github.com/PetarV-/TikZ/tree/master/2D%20Convolution (accessed on 2022-06-14). 16

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008. 7, 22, 63, 69

Wolfgang Wahlster and Christoph Winterhalter. German standardization roadmap on artificial intelligence, 2020. URL https://www.din.de/resource/blob/772610/e96c34dd6b12900ea75b460538805349/normungsroadmap-en-data.pdf. (accessed on 2022-01-12). 94

Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (eds.), *Computer Vision - ECCV 2016 - 14th*

European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VII, volume 9911 of *Lecture Notes in Computer Science*, pp. 835–851. Springer, 2016. doi: 10.1007/978-3-319-46478-7_51. URL https://doi.org/10.1007/978-3-319-46478-7_51. 39

Izhar Wallach, Michael Dzamba, and Abraham Heifets. AtomNet: A deep convolutional neural network for bioactivity prediction in structure-based drug discovery. *CoRR*, abs/1510.02855, 2015. URL http://arxiv.org/abs/1510.02855. 17

Li Wan, Matthew D. Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. Regularization of neural networks using DropConnect. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pp. 1058–1066. JMLR.org, 2013. URL http://proceedings.mlr.press/v28/wan13.html. 25

Bin Wang, Angela Wang, Fenxiao Chen, Yuncheng Wang, and C-C Jay Kuo. Evaluating word embedding models: Methods and experimental results. *APSIPA Transactions on Signal and Information Processing*, 8, 2019. 42

Jie Wang. An intuitive tutorial to Gaussian processes regression. *CoRR*, abs/2009.10862, 2020. URL https://arxiv.org/abs/2009.10862. Source code is available at https://github.com/jwangjie/Gaussian-Processes-Regression-Tutorial (accessed on 2022-03-01). 30

Greg Welch and Gary Bishop. An introduction to the Kalman filter. Technical Report 95-041, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1995. URL http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html. 117

Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient Langevin dynamics. In Lise Getoor and Tobias Scheffer (eds.), *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pp. 681–688, 2011. URL https://icml.cc/2011/papers/398_icmlpaper.pdf. 36, 112

Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=Sklf1yrYDr. 38

Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/481fbfa59da2581098e841b7afc122f1-Abstract.html. 38

Paul Werbos. *Beyond regression: new tools for prediction and analysis in the behavioral sciences*. PhD thesis, Harvard University, 1974. 13

Oliver Willers, Sebastian Sudholt, Shervin Raafatnia, and Stephanie Abrecht. Safety concerns and mitigation approaches regarding the use of deep learning in safety-critical perception tasks. In António Casimiro, Frank Ortmeier, Erwin Schoitsch, Friedemann Bitsch, and Pedro M. Ferreira (eds.), *Computer Safety, Reliability, and Security. SAFECOMP 2020 Workshops - DECSoS 2020, DepDevOps 2020, USDAI 2020, and WAISE 2020, Lisbon, Portugal, September 15, 2020, Proceedings*, volume 12235 of *Lecture Notes in Computer Science*, pp. 336–350. Springer, 2020. doi: 10.1007/978-3-030-55583-2_25. URL https://doi.org/10.1007/978-3-030-55583-2_25. 95

Sascha Wirges, Marcel Reith-Braun, Martin Lauer, and Christoph Stiller. Capturing object detection uncertainty in multi-layer grid maps. In *2019 IEEE Intelligent Vehicles Symposium, IV 2019, Paris, France, June 9-12, 2019*, pp. 1520–1526. IEEE, 2019. doi: 10.1109/IVS.2019.8814073. URL https://doi.org/10.1109/IVS.2019.8814073. 40

Magnus Wrenninge and Jonas Unger. Synscapes: A photorealistic synthetic dataset for street scene parsing. *CoRR*, abs/1810.08705, 2018. URL http://arxiv.org/abs/1810.08705. 84

W. A. Wright, Guillaume Ramage, Dan Cornford, and Ian T. Nabney. Neural network modelling with input uncertainty: Theory and application. *J. VLSI Signal Process.*, 26(1-2):169–188, 2000. doi: 10.1023/A:1008111920791. URL https://doi.org/10.1023/A:1008111920791. 101

Anqi Wu, Sebastian Nowozin, Edward Meeds, Richard E. Turner, José Miguel Hernández-Lobato, and Alexander L. Gaunt. Deterministic variational inference for robust Bayesian neural networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=B1l08oAct7. 54

Bichen Wu, Forrest N. Iandola, Peter H. Jin, and Kurt Keutzer. SqueezeDet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 446–454. IEEE Computer Society, 2017. doi: 10.1109/CVPRW.2017.60. URL https://doi.org/10.1109/CVPRW.2017.60. 7, 17, 62, 82, 83, 128

Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Masayoshi Tomizuka, Kurt Keutzer, and Peter Vajda. Visual transformers: Token-based image representation and processing for computer vision. *CoRR*, abs/2006.03677, 2020a. URL https://arxiv.org/abs/2006.03677. 17

CF Jeff Wu. On the convergence properties of the EM algorithm. *The Annals of Statistics*, pp. 95–103, 1983. 42

Xiongwei Wu, Doyen Sahoo, and Steven CH Hoi. Recent advances in deep learning for object detection. *Neurocomputing*, 2020b. 42

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 55

Yifei Xue, Tiejun Wang, and Andrew K. Skidmore. Automatic counting of large mammals from very high resolution panchromatic satellite imagery. *Remote. Sens.*, 9(9):878, 2017. doi: 10.3390/rs9090878. URL `https://doi.org/10.3390/rs9090878`. 82

Fanny Yang, Sivaraman Balakrishnan, and Martin J Wainwright. Statistical and computational guarantees for the Baum-Welch algorithm. *The Journal of Machine Learning Research*, 18(1):4528–4580, 2017. 42

Yu Yao, Mingze Xu, Yuchen Wang, David J. Crandall, and Ella M. Atkins. Unsupervised traffic accident detection in first-person videos. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2019, Macau, SAR, China, November 3-8, 2019*, pp. 273–280. IEEE, 2019. doi: 10.1109/IROS40897.2019.8967556. URL `https://doi.org/10.1109/IROS40897.2019.8967556`. 121

Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. BDD100K: A diverse driving dataset for heterogeneous multitask learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 2633–2642. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.00271. URL `https://openaccess.thecvf.com/content_CVPR_2020/html/Yu_BDD100K_A_Diverse_Driving_Dataset_for_Heterogeneous_Multitask_Learning_CVPR_2020_paper.html`. 84

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Richard C. Wilson, Edwin R. Hancock, and William A. P. Smith (eds.), *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. BMVA Press, 2016. URL `http://www.bmva.org/bmvc/2016/papers/paper087/index.html`. 35

Sheheryar Zaidi, Arber Zela, Thomas Elsken, Chris C. Holmes, Frank Hutter, and Yee Whye Teh. Neural ensemble search for uncertainty estimation and dataset shift. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 7898–7911, 2021. URL `https://proceedings.neurips.cc/paper/2021/hash/41a6fd31aa2e75c3c6d427db3d17ea80-Abstract.html`. 38

Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. URL `http://arxiv.org/abs/1212.5701`. 19

Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8660–8669, 2019. 1

Aonan Zhang, San Gultekin, and John Paisley. Stochastic variational inference for the HDP-HMM. In *Artificial Intelligence and Statistics*, pp. 800–808, 2016a. 44

Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, pp. 1–1, 2020a. doi: 10.1109/TSE.2019.2962027. 95, 96

Jing Zhang and Jun Zhuang. Modeling a multi-target attacker-defender game with multiple attack types. *Reliab. Eng. Syst. Saf.*, 185:465–475, 2019. doi: 10.1016/j.ress.2019.01.015. URL `https://doi.org/10.1016/j.ress.2019.01.015`. 126

Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical stochastic gradient MCMC for Bayesian deep learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020b. URL `https://openreview.net/forum?id=rkeS1RVtPS`. 37

Ye Zhang, Md. Mustafizur Rahman, Alex Braylan, Brandon Dang, Heng-Lu Chang, Henna Kim, Quinten McNamara, Aaron Angert, Edward Banner, Vivek Khetan, Tyler McDonnell, An Thanh Nguyen, Dan Xu, Byron C. Wallace, and Matthew Lease. Neural information retrieval: A literature review. *CoRR*, abs/1611.06792, 2016b. URL `http://arxiv.org/abs/1611.06792`. 42

Shengjia Zhao, Tengyu Ma, and Stefano Ermon. Individual calibration with randomized forecasting. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 11387–11397. PMLR, 2020. 105

Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Trans. Neural Networks Learn. Syst.*, 30(11):3212–3232, 2019. doi: 10.1109/TNNLS.2018.2876865. URL `https://doi.org/10.1109/TNNLS.2018.2876865`. 82

Li-Qiang Zhou, Jia-Yu Wang, Song-Yuan Yu, Ge-Ge Wu, Qi Wei, You-Bin Deng, Xing-Long Wu, Xin-Wu Cui, and Christoph F Dietrich. Artificial intelligence in medical imaging of the liver. *World Journal of Gastroenterology*, 25(6):672, 2019. 1

Quan Zou, Pengwei Xing, Leyi Wei, and Bin Liu. Gene2vec: gene subsequence embedding for prediction of mammalian N6-methyladenosine sites from mRNA. *RNA*, 25(2): 205–218, 2019. 42