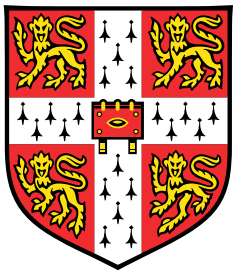
Uncertainty Estimation in Deep Learning with application to Spoken Language Assessment



Andrey Malinin

Supervisor: Professor Mark J. F. Gales

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Christ's College

August 2019

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Andrey Malinin
August 2019

Abstract

Since convolutional neural networks (CNNs) achieved top performance on the ImageNet task in 2012, deep learning has become the preferred approach to addressing computer vision, natural language processing, speech recognition and bio-informatics tasks. However, despite impressive performance, neural networks tend to make over-confident predictions. Thus, it is necessary to investigate robust, interpretable and tractable estimates of uncertainty in a model's predictions in order to construct safer Machine Learning systems. This is crucial to applications where the cost of an error is high, such as in autonomous vehicle control, high-stakes automatic proficiency assessment and in the medical, financial and legal fields.

In the first part of this thesis uncertainty estimation via ensemble and single-model approaches is discussed in detail and a new class of models for uncertainty estimation, called *Prior Networks*, is proposed. Prior Networks are able to *emulate* an ensemble of models using a single deterministic neural network, which allows sources of uncertainty to be determined within the same probabilistic framework as in ensemble-based approaches, but with the computational simplicity and ease of training of single-model approaches. Thus, Prior Networks combine the advantages of ensemble and single-model approaches to estimating uncertainty. In this thesis Prior Networks are evaluated on a range classification datasets, where they are shown to outperform baseline approaches, such as Monte-Carlo dropout, on the task of detecting out-of-distribution inputs.

In the second part of this thesis deep learning and uncertainty estimation approaches are applied to the area of automatic assessment of non-native spoken language proficiency. Specifically deep-learning based graders and spoken response relevance assessment systems are constructed using data from the BULATS and LinguaSkill exams, provided by Cambridge English Language Assessment. Baseline approaches for uncertainty estimation discussed and evaluated in the first half of the thesis are then applied to these models and assessed on the task of rejecting predictions to be graded by human examiners and detecting misclassifications.

Acknowledgements

First and foremost, I would like to thank my supervisor, Professor Mark Gales, for his guidance, maverick insight, enthusiasm and patience. I am grateful that he has always made time to address my questions and has provided honest, candid, and wise advice. Secondly, I would like to thank (in alphabetical order) Kate Knill, Konstantinos Kyriakopoulos, Anton Ragni and Yu Wang for invaluable comments and discussions during the writing of various papers and this thesis. Additionally, I would like to thank Bruno Mlodozeniec for his assistance in writing the code for work in chapter 8 of this thesis as well as help with visualizations for talks I have given. I would also like to thank Ramón Fernández Astudillo, Nicholas Carlini, José Miguel Hernández-Lobato, Richard Turner, Dmitry Vetrov and Mark van der Wilk for thought provoking and insightful discussions which have helped me gain a wider view of machine learning. Many thanks must also go to my friends, and especially my parents, for their continued support. Lastly, I would like to thank Cambridge English and EPSRC for supporting five years of my PhD and providing real data from spoken language proficiency exams, which was used in the second half of this thesis.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

ATM	Attention-based Model
BiLSTM	Bidirectional LSTM
BNN	Bayesian Neural Network
CE	Cross-Entropy
DNN	Deep Neural Network
DPN	Dirichlet Prior Network
EPKL	Expected Pairwise KL-Divergence
HATM	Hierarchical Attention-based Model
KL	Kullback-Leibler Divergence
LM	Language Model
LSTM	Long Short-Term Memory Recurrent Neural Network
MI	Mutual Information
NLL	Negative Log-Likelihood
NPN	Normal-inverse-Wishart Prior Network
OOD	Out-of-distribution / Out-of-domain
PN	Prior Network
RCE	Reverse Cross-Entropy

RKL Reverse Kullback-Leibler Divergence

RNN Recurrent Neural Network

RNNLM Recurrent Neural Network Language Model

SNN Siamese Neural Network

Greek Symbols

$\boldsymbol{\alpha}$ Concentration Parameters of Dirichlet

$\boldsymbol{\beta}$ target Concentration Parameters of Dirichlet

$\boldsymbol{\mu}$ Mean

$\boldsymbol{\pi}$ Parameters of categorical distribution

$\boldsymbol{\Sigma}$ Covariance

$\boldsymbol{\theta}$ Model Parameters

η Learning Rate

γ OOD loss weight

κ Belief strength in $\mathcal{NW}^{-1}$ Prior Mean

ν Belief strength in $\mathcal{NW}^{-1}$ Prior Scatter Matrix

ω Discrete Class

Measures of Uncertainty

$\mathcal{H}[\cdot]$ Entropy or Differential Entropy

$\mathcal{I}[\cdot]$ Mutual Information (note square brace)

$\mathcal{K}[\cdot]$ Expected Pairwise KL-divergence

Other Symbols

$\mathcal{R}$ Real Numbers

Probability Distributions

$\mathbb{E}$ Expectation

$\mathbb{V}$	Variance
$\mathcal{B}$	Bernoulli Distribution
$\mathcal{NW}^{-1}$	Normal inverse-Wishart Distribution
$\mathcal{N}$	Normal Distribution
$\mathcal{W}^{-1}$	inverse-Wishart Distribution
`Cat`	Categorical Distribution
`Dir`	Dirichlet Distribution
$\mathtt{p}(\boldsymbol{y})$	Distribution over continuous random variable
$\mathtt{P}(y)$	Distribution over discrete random variable

Roman Symbols

$\boldsymbol{h}$	Vector representation or neural network hidden state
$\boldsymbol{m}$	Prior Mean of Normal-inverse-Wishart distribution
$\boldsymbol{S}$	Prior Scatter Matrix of Normal-inverse-Wishart distribution
$\boldsymbol{W}$	Set of word sequences
$\boldsymbol{w}$	Word-sequence
$\boldsymbol{x}$	Input
$\boldsymbol{x}^*$	Input from test dataset
$\boldsymbol{y}$	Continuous target
$\mathcal{D}$	Dataset
$\mathcal{I}(\cdot)$	Indicator function (note normal brackets)
$\mathcal{L}$	Loss
$\mathcal{M}$	Model (generalization of model parameters)
y	Discrete target

Chapter 1

Introduction

Artificial Neural Networks are models which are loosely inspired by the human visual system. The first Neural Network (NN) model, the *Perceptron* was constructed in 1955 [106]. Since then numerous developments have occurred, most importantly, the discovery of the *Back Propagation* algorithm [107] in the 1980s, which allows neural networks to be efficiently trained using gradient-descent based methods. The current wave of progress in machine learning began in 2012 when convolutional neural networks (CNNs) achieved top performance on the ImageNet task [66]. Since then *deep learning*, the subset of machine learning which deals with neural networks, has become the preferred approach to addressing computer vision (CV) [37, 113, 125], natural language processing (NLP) [90, 89, 87], speech recognition (ASR) [53, 47] and bio-informatics (BI) [17, 5] tasks.

Despite impressive, and ever-improving performance, neural networks tend to make over-confident predictions [68, 95]. Until recently, there has been little work done on deriving, understanding, using and evaluating estimates of uncertainty[1] in the predictions of neural networks, beyond simple measures like confidence scores [32]. However, estimating uncertainty in a model's predictions is important in many practical applications, as it enables, for example, the safety of an Artificial Intelligence (AI) / Machine Learning (ML) system [6] to be increased by acting on the model's prediction in an informed manner. This is crucial to applications where the cost of an error is high, such as in autonomous vehicle control, high-stakes automatic proficiency assessment and in the medical, financial and legal fields.

To illustrate the need for good uncertainty estimates, let's consider two hypothetical scenarios involving an autonomous self-driving car. Firstly, consider the situation where the car's AI has been trained on data collected in California, where the weather and road

[1] There are a range of definitions of 'uncertainty' in the literature. In this thesis uncertainty will refer to 'uncertainty in a discriminative model's predictions'. In other words uncertainty is the inverse of confidence in a discriminative prediction.

conditions are good. However, the car is driven to Cambridge, where the cars drive on the left, the roads are narrow, the weather is poor and traffic signs are different. The conditions of deployment differ significantly from the data on which it was trained, which may lead the car's AI to make poor driving decisions. In this situation, a safe course of action would be to realize that the conditions differ from the training data and hand over control to a human driver. The car's on-board AI should have an understanding of the limits of its knowledge and when it is uncertain in the actions it takes. Now let's consider an alternative situation where the car's cameras and other sensors get obscured by heavy rain and street signs are covered with mud, for example. However, unlike before, these conditions have been encountered in training data and the on-board AI may understand that it is impossible to make safe driving decisions in these conditions. Here, the car should either hand over control to the human passenger or initiate a sensor cleaning procedure, which is akin to a human driver turning on the windshield wipers or switching on fog-lights.

The first hypothetical scenario is an example of "uncertainty in predictions due to lack of knowledge about or understand of the current input data", which shall be referred to as *knowledge uncertainty*. *Knowledge uncertainty* is a form of *unknown-unknown* - the AI lacks the knowledge and understanding to make sense of the data, as it is like nothing it has encountered in the training data. The second scenario is an example of "uncertainty in predictions due to uncertainty or noise in the data", which will be referred to as *data uncertainty*. *Data uncertainty* can be considered a *known-unknown* - the AI has encountered situations like this in the training data and understands that taking appropriate actions for these inputs is impossible. *Knowledge uncertainty* and *data uncertainty* are the two *sources of uncertainty* in predictions. This distinction is important, as in certain applications of machine learning it may be necessary to know not only *whether* the model is uncertain, but also *why*.

A real-life example illustrating the need for uncertainty estimates is when an Irish veterinarian living in Australia, who spoke perfect native English, failed an automatically assessed spoken English proficiency exam [1] which was used to assess candidates applying for permanent residence. In this situation, it is likely that the automatic assessment system was trained on Australian speakers of English and the significant accent mismatch caused the system to incorrectly fail the Irish veterinarian. Had this system been able to provide estimates of uncertainty, it could have understood that there was a mismatch between training and deployment data, an example of *knowledge uncertainty*, and deferred to a human assessment. Cases like this illustrate how AI systems making potentially life-changing high-stakes decisions must yield robust estimates of uncertainty in their predictions in order to avoid mistakes.

Fortunately, notable progress has been recently made on predictive uncertainty estimation for neural networks through the definition of baselines, tasks and metrics [49, 68] and the development of a range of practical methods for estimating uncertainty. As will be shown in chapter 3, *data uncertainty* can be captured by probabilistic classification and regression models as a consequence of maximum likelihood estimation. Modelling *knowledge uncertainty* is more difficult, however.

One class of approaches are *Bayesian Neural Networks* [79, 78, 55, 94, 35, 28]. Here, a prior distribution over model parameters $\mathrm{p}(\boldsymbol{\theta})$ is used to obtain a posterior distribution over model parameters conditioned on the training data $\mathrm{p}(\boldsymbol{\theta}|\mathcal{D})$. The posterior captures the *uncertainty in model parameters*, which can be used to derive estimates of the *uncertainty in the predictions due to uncertainty in model parameters* which will be referred to as *model uncertainty*. Given an appropriate choice of prior and model class, *model uncertainty* will capture *knowledge uncertainty*. Furthermore, Bayesian approaches also allow the sources of uncertainty to be determined [35, 28]. However, Bayesian Neural Networks (BNNs) have traditionally been computationally more demanding and conceptually more complicated than non-Bayesian NNs. Since exact Bayesian inference is intractable for neural network models, the performance of Bayesian Neural Networks depends on the form of approximations made due to computational constraints and the nature of the prior distribution over parameters. Typically inference is done by considering Monte-Carlo sampling from an approximate posterior $\mathrm{q}(\boldsymbol{\theta})$. As a consequence, these methods can be interpreted as a form of ensemble approach. A recent development has been the technique of Monte-Carlo Dropout [36, 35], which approximates Monte-Carlo sampling from the posterior by doing multiple stochastic forward passes through a neural network. Uncertainty estimates are derived by computing the mean and spread of an ensemble of models sampled using this approach. While work by [98] suggests that Monte-Carlo Dropout does not sample from an appropriate Bayesian posterior, Monte-Carlo Dropout has nevertheless been successfully applied to tasks in computer vision [61, 60]. A number non-Bayesian ensemble approaches have also been proposed, which also allow measures of uncertainty to be derived from the spread of an ensemble. An example of this class of ensemble approaches is Deep Ensembles [68], which involves explicitly training an ensemble of neural networks and has been shown to provide estimates of uncertainty competitive with Monte Carlo Dropout. Instead of using ensembles of models to capture *knowledge uncertainty*, it is possible to consider an alternative class of approaches which involves only a point-estimate of model parameters $\boldsymbol{\theta}$, which will be referred to as *single model approaches*. While there is a wide range of different single model approaches to capturing *knowledge uncertainty* [74, 80, 73, 91, 49] which yield good results, many do not have a good theoretical justification. Thus, in this thesis only a specific type of single

model approach is considered, where a neural network is *explicitly* trained in a multi-task fashion to yield low or high estimates of uncertainty for inputs which are near to, or far from, the training data, respectively [80, 73]. The performance of these approaches depends on choice of *out-of-domain training data*, which can either be sampled from a synthetic noise distribution or taken from a different dataset during training. These approaches are both simpler and more computationally efficient both at training and test time than Bayesian ensemble approaches. However, unlike ensemble methods, they do not allow the source of uncertainty to be determined within a non-heuristic, theoretically consistent probabilistic framework. This makes it difficult to use them for tasks where it is important to determine the source of uncertainty, such as active learning [112], for example.

Summary of Contributions. The contributions of this thesis are broadly split into two parts. In the first part ensemble and single model approaches are discussed in detail and a new class of models for uncertainty estimation called Prior Networks (PNs) is proposed. Prior Networks aim to combine the advantages of ensemble and single-model approaches to estimating uncertainty in predictions. Specifically, Prior Networks are a single model approach to estimating uncertainty which allows sources of uncertainty to be determined within the same probabilistic framework as in ensemble-based approaches, but with the computational simplicity and ease of training of single-model approaches. Prior Networks are evaluated on a range image classification datasets where they are used for the tasks of misclassification detection and out-of-distribution sample detection.

As previously stated, it is important for automatic proficienct assessment systems to yield robust estimates of uncertainty. Thus, the second part of this thesis involves applying deep learning approaches and uncertainty estimation to the area of automatic spoken language assessment. Firstly, deep-learning based systems for automatic assessment of non-native spoken language proficiency are developed. These systems are then evaluated on examination candidates' spoken responses to question-prompts on the BULATS [15] exam provided by Cambridge English Language Assessment. Approaches for uncertainty estimation discussed and evaluated in the first half of the thesis are then applied to these models and assessed on the task of rejecting predictions to be graded by human examiners. Secondly, deep-learning based systems for automatic assessment of relevance of spoken responses to open-ended exam prompts are developed and applied to the BULATS [15] and LinguaSkill [2] exams. Ensemble approaches to uncertainty estimation were applied to these models and used to detect on which prompts and responses the system made the biggest errors.

1.1 Thesis Structure

The structure of this thesis is as follows: chapter 2 introduces the underlying theory on deep learning, maximum likelihood estimation and gradient descent optimization necessary to support this thesis. Chapters 3-5 are the theoretical and experimental contributions to uncertainty estimation for deep learning. Chapter 6 is a transition chapter which introduces the area of spoken language proficiency assessment and discusses the associated challenges. Chapters 7 and 8 investigate the application of deep learning and uncertainty estimation to automatic grading of spoken language proficiency and the assessment of the relevance of spoken responses to open-ended exam prompts. Chapter 9 concludes the thesis with an overview of each chapter and a summary of proposed future work. In more detail:

Chapter 2 discusses theory which is necessary to support discussions throughout the rest of the thesis. This chapter discusses deep learning and standard neural network architectures, such as feed-forward, convolutional, recurrent and attention layers, maximum likelihood estimation and gradient-descent based optimization.

Chapter 3 introduces the area of predictive uncertainty estimation. The sources of uncertainty - *data uncertainty* and *knowledge uncertainty* are discussed in the context of classification and regression tasks. It is shown how probabilistic classification and regression models capture *data uncertainty* as a consequence of maximum likelihood estimation. Finally, single-model and ensemble approaches to capturing *knowledge uncertainty* are discussed and contrasted.

Chapter 4 describes the main theoretical contribution of this thesis - a new class of models for uncertainty estimation, called *Prior Networks*, which combine advantages of single model and ensemble approaches to estimating uncertainty. Specifically, Prior Networks are a single model approach that operates within the same probabilistically coherent framework as ensemble approaches. This allows the sources of uncertainty to be determined using the same measures and decompositions as ensemble approaches, but at the computational expense of single-model approaches.

Chapter 5 is an experimental chapter in which Prior Networks are evaluated on a set of image recognition tasks. Specifically, the uncertainty estimates produced by Prior Networks trained on the MNIST [70], SVHN [41] and CIFAR-10 [41] datasets are evaluated on two applications of uncertainty: detection of misclassifications and detection of out-of-distribution inputs. Approaches discussed in Chapter 3 are used as baselines to compare Prior Networks against.

Chapter 6 transitions from the first part of this thesis to the second. It discusses the area of assessment of spoken language proficiency and introduces the tasks of automatic grading and prompt-response relevance assessment, which are investigated in chapters 7

and 8. Furthermore, it details the BULATS and LinguaSkill spoken language proficiency exams. Automatic graders and prompt-response relevance assessment models will be trained on spoken responses of candidate taking these exams in chapters 7 and 8.

Chapter 7 applies deep learning to the task of automatic grading of spoken language proficiency. Models are trained and evaluated on data from the BULATS exam introduced in chapter 6. Performance of models based on Density Networks, discussed in chapter 3, is compared to the performance of Gaussian Process graders. Additionally, estimates of uncertainty in predictions, derived using approaches introduced in chapter 3, are evaluated on the task of rejecting automatic predictions to be re-assessed by human graders.

Chapter 8 applies deep learning to the task of assessing the relevance of spoken responses to question-prompts on the BULATS and LinguaSkill exams. There has been little past work in this area, so there are no 'standard' baseline models, unlike in chapters 5 and 7. Two classes of approaches to prompt-response relevance assessment are considered - indirect approaches, which treat relevance assessment as a prompt-classification task, and direct approaches, where models directly yield a relevance score for a prompt-response pair. Estimates of uncertainty in predictions, derived using approaches introduced in chapter 3, are applied to direct relevance assessment models and are then evaluated on two tasks. Firstly, estimates of uncertainty are evaluated on the misclassification detection task described in Chapter 5. Secondly, uncertainty estimates are used to reject automatic predictions of prompt-response relevance to be re-assessed by human assessors.

Chapter 9 concludes the thesis with an overview of the contributions of each chapter and a summary of proposed future work.

Chapter 2

Deep Learning

This chapter introduces theory needed to support discussions in the rest of the thesis. The chapter is structured into sections discussing neural networks and deep learning, maximum likelihood estimation, and optimization via stochastic gradient-descent. Specifically, section 2.1 introduces neural networks and deep learning, and discusses standard neural network architectures, such as fully-connected, convolutional, recurrent and attention layers. Section 2.2 discusses training of probabilistic models parameterized by neural networks via maximum likelihood estimation. This discussion also motivates the need for regularization. Section 2.3 discusses stochastic gradient descent and details several optimizers, such as Adam [62]. Additionally, this section also discusses the issue of initialization of neural networks.

2.1 Deep Neural Networks

Neural Networks are non-linear functions with a multi-layer structure which have been applied to construct models for addressing computer vision (CV) [37, 113, 125], natural language processing (NLP) [90, 89, 87], speech recognition (ASR) [53, 47] and bio-informatics (BI) [17, 5] tasks. Neural networks are nonlinear functions parameterized by parameters $\boldsymbol{\theta}$:

$$\hat{\boldsymbol{y}} = f(\boldsymbol{x}; \boldsymbol{\theta}) \tag{2.1}$$

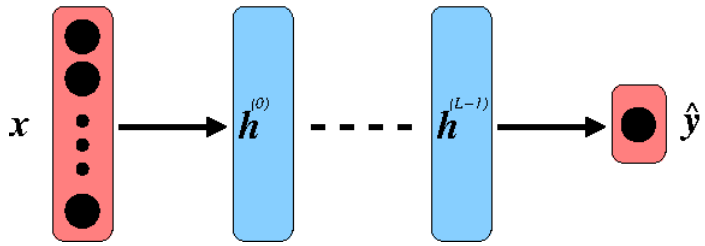


Figure 2.1 Feed-Forward Neural Network

In general, neural networks are structured as a sequence of several nonlinear transformations or *layers*:

$$\begin{aligned} \boldsymbol{h}^{(0)} &= f(\boldsymbol{x}; \boldsymbol{\theta}^{(0)}) \\ \boldsymbol{h}^{(l)} &= f(\boldsymbol{h}^{(l-1)}; \boldsymbol{\theta}^{(l)}) \\ \hat{\boldsymbol{y}} &= f(\boldsymbol{h}^{(L-1)}; \boldsymbol{\theta}^{(L)}) \end{aligned} \tag{2.2}$$

Each of these transformations is called a *hidden layer* and the output of each hidden layer $\boldsymbol{h}^{(l)}$ is called the *hidden state*. A neural network with no hidden layers can only learn linear relationships and model linear decision boundaries. If one hidden layer is used then the network is called a *Multi-Layer Perceptron*. Multi-Layer Perceptrons are universal function approximators [11, 92, 40] - given a hidden layer which is sufficiently large it can model arbitrary functions of the input. Models which have more than one hidden layer are known as Deep Neural Networks (DNNs) and are considered to be more *efficient* function approximators than an MLP. This is known as the *deep learning hypothesis* [9]. A visual representation of a deep neural network is given in figure 2.1. Neural networks can be seen as powerful, learnable, non-linear feature extractors which project the input $\boldsymbol{x}$ into a space where they can be successfully modelled using linear regression or classification models [11, 40].

While there exists a great variety of different hidden layers and ways in which they can be connected, this section will cover only the most standard ones. The rest of this section discusses common *feed-forward* and *recurrent* neural networks. Section 2.1.1 discusses fully connected layers (FNNs), which are the simplest type of layer, and convolutional layers (CNNs), which are the basic building blocks of neural networks for image processing. Furthermore, common *activation* and *output* functions are also discussed in this section. Section 2.1.2 discusses both basic recurrent neural network layers (RNNs) as well as sophisticated *Long Short-Term Memory* (LSTM) recurrent neural networks, which are currently one of the default architectures for sequence modelling neural networks. Section 2.1.3 discusses neural attention mechanisms, which are also an important component of many

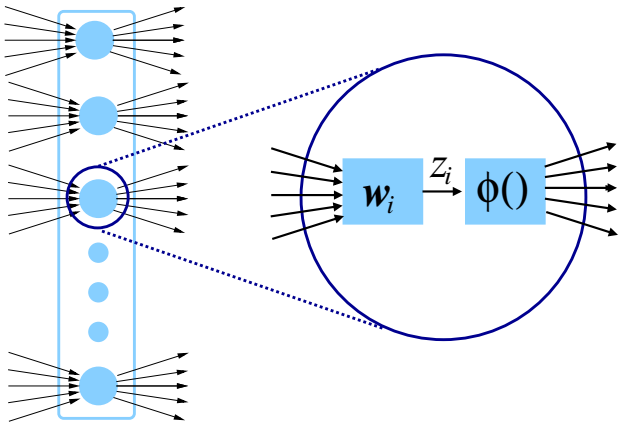


Figure 2.2 Structure of a fully connected hidden layer

sequence-modelling networks. Finally, section 2.1.4 discusses how neural networks can be used to parameterize distributions over discrete and continuous random variables.

2.1.1 Feed-Forward Neural Networks

Feed-forward neural networks are non-linear functions which model mappings from an arbitrary input $\boldsymbol{x}$ to an arbitrary output $\boldsymbol{y}$:

$$\boldsymbol{x} \mapsto \boldsymbol{y} \tag{2.3}$$

Even though the input $\boldsymbol{x}$ may have *internal structure*, in this thesis these mappings will be considered *unstructured* because the mapping itself is simply between arbitrary vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.

The simplest type of feed-forward layer is the *fully connected* layer, depicted in figure 2.4, which consists of an affine transformation of the input followed by an *element-wise* non-linear *activation function* $\phi(\cdot)$:

$$\begin{aligned} \boldsymbol{z}^{(l)} &= \boldsymbol{W}^{(l)}\boldsymbol{h}^{(l-1)} + \boldsymbol{b}^{(l)} \\ \boldsymbol{h}^{(l)} &= \phi(\boldsymbol{z}^{(l)}) \\ \boldsymbol{\theta}^{(l)} &= \{\boldsymbol{W}^{(l)}, \boldsymbol{b}^{(l)}\} \end{aligned} \tag{2.4}$$

where $\boldsymbol{W}^{(l)}$ is a $H^{(l-1)} \times H^{(l)}$ weight matrix and $\boldsymbol{b}^{(l)}$ is a $H^{(l)}$ dimensional bias vector. Typical choices of activation function are the *logistic* or *sigmoid* function, the *hyperbolic*

tangent (tanh) function, the *Recitified Linear* and the *Leaky Rectified Linear* functions:

$$\begin{aligned}
\phi_{\texttt{sigmoid}}(x) &= \frac{1}{1+e^{-x}} \\
\phi_{\texttt{tanh}}(x) &= \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \\
\phi_{\texttt{ReLU}}(x) &= \max(0,x) \\
\phi_{\texttt{LReLU}}(x) &= \max(\alpha \cdot x, x),\ 0.0 < \alpha < 1.0
\end{aligned}$$

In this thesis, all neural networks use Leaky ReLU activation functions in their hidden layers.

Feed-forward networks designed especially for image processing are called *Convolution Neural Networks* [71]. CNNs use *convolutional layers* to build in invariances to translation and incorporate parameter sharing. Effectively, CNNs contain a strong *structural prior* on how image data should be processed. A CNN represents the activations of the l-th hidden layer as a 3D tensor $\boldsymbol{H}^{(l)} \in \mathcal{R}^{H^{(l-1)} \times W^{(l-1)} \times F^{(l-1)}}$ with height $H^{(l-1)}$, width $W^{(l-1)}$ and $F^{(l-1)}$ channels. Each 2D $H^{(l-1)} \times W^{(l-1)}$ slice of this tensor is a *feature map* or channel. The weights per layer are no longer simple matrices (though they could be represented as such), but rather a set of $K^{(l)}$ *kernels* $\boldsymbol{K} \in \mathcal{R}^{h \times w \times K^{(l)}}$. The output of each layer is the convolution of each kernel with each filter map, followed by an element-wise non-linear activation function:

$$\boldsymbol{H}_j^{(l)} = \phi\Big(\sum_{f \in F^{(l-1)}} \boldsymbol{H}_f^{(l-1)} * \boldsymbol{K}_j^{(l)} + \boldsymbol{b}_j^{(l)}\Big) \quad \forall j \in F^{(l)} \tag{2.5}$$

Currently, there is a wide range of 'standard' convolutional neural network architectures for the task of image classification, such as the VGG architecture [113], Residual Network [48] and DenseNets [59]. However, in this thesis only VGG-style CNNs are considered.

2.1.2 Recurrent Neural Networks

Feed-forward neural networks capture *unstructured* vector-to-vector mappings $\boldsymbol{x} \mapsto \boldsymbol{y}$. However, it is possible to use neural networks to model arbitrary sequence-to-vector, vector-to-sequence and sequence-to-sequence mappings:

$$\begin{aligned}
\{\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(T)}\} &\mapsto \boldsymbol{y} \\
\boldsymbol{x} &\mapsto \{\boldsymbol{y}^{(1)}, \cdots, \boldsymbol{y}^{(\tau)}\} \\
\{\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(T)}\} &\mapsto \{\boldsymbol{y}^{(1)}, \cdots, \boldsymbol{y}^{(\tau)}\}
\end{aligned} \tag{2.6}$$

In this thesis these mappings will be referred to as *structured*.

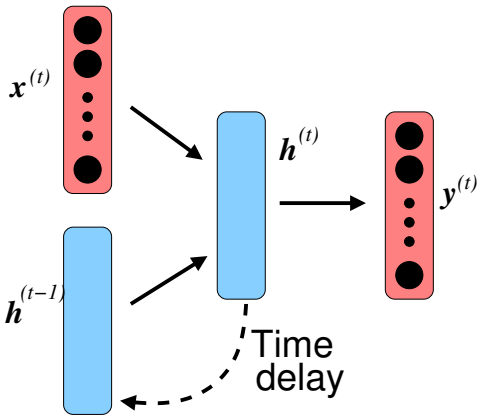


Figure 2.3 Recurrent Neural Network

Mappings to and from sequences can be modelled using a *Recurrent Neural Network* (RNN), which is depicted in figure 2.3. RNNs have a recurrent 'hidden' state $\boldsymbol{h}^{(t)}$ which encapsulates all past inputs in a single vector. This hidden state is a function of the current input and the hidden state at the past time step $\boldsymbol{h}^{(t-1)}$. The output of the RNN at every time step is a non-linear function of the hidden state $\boldsymbol{h}^{(t)}$:

$$\begin{aligned} \boldsymbol{h}^{(t)} &= \phi_{\texttt{tanh}}(\boldsymbol{W}^{(r)}\boldsymbol{h}^{(t-1)} + \boldsymbol{W}^{(i)}\boldsymbol{x}^{(t)} + \boldsymbol{b}^{(r)}) \\ \boldsymbol{y}^{(t)} &= \phi(\boldsymbol{W}^{(o)}\boldsymbol{h}^{(t)} + \boldsymbol{b}^{(o)}) \end{aligned} \tag{2.7}$$

where $\boldsymbol{W}^{(i)}$ is the input weight matrix, $\boldsymbol{W}^{(r)}$ is the recurrent hidden state transition matrix and $\boldsymbol{W}^{(o)}$ is the output weight matrix, and $\boldsymbol{b}^{(r)}$ and $\boldsymbol{b}^{(o)}$ are bias vectors. RNNs can represent sequence-to-vector relationships by processing a sequence of inputs $\{\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(T)}\}$ and only outputting the final $\boldsymbol{y}^{(T)}$. This can easily be expanded to model sequence-to-sequence relationships by reading out each $\boldsymbol{y}^{(t)}$. RNNs are extremely flexible and many other forms of sequence-to-vector, sequence-to-sequence and vector-to-sequence relationships can be constructed.

A limitation of RNNs as presented in equation 2.7 is that the content of the hidden state is completely overwritten on every timestep. In practice it is hard to control what the RNN remembers and what it does not - which information is retained for a long time and which is forgotten after only a few time steps. To overcome this limitation, a very popular *gated architecture* called the *Long Short-Term Memory* (LSTM) RNN was introduced [57, 136, 42]. The idea behind an LSTM is to maintain a *cell state* $\boldsymbol{c}^{(t)}$ which serves as a perfect memory cell. Three 'gates' - the input, forget and output gates are added which control information flow to and from the memory cell. The gates are functions of the current input $\boldsymbol{x}^{(t)}$ and the

past output hidden state $\boldsymbol{h}^{(t-1)}$:

$$\begin{aligned}
\boldsymbol{i} &= \phi_{\texttt{sigmoid}}(\boldsymbol{U}^{(i)}\boldsymbol{h}^{(t-1)} + \boldsymbol{W}^{(i)}\boldsymbol{x}^{(t)} + \boldsymbol{b}^{(i)}) \\
\boldsymbol{f} &= \phi_{\texttt{sigmoid}}(\boldsymbol{U}^{(f)}\boldsymbol{h}^{(t-1)} + \boldsymbol{W}^{(f)}\boldsymbol{x}^{(t)} + \boldsymbol{b}^{(f)}) \\
\boldsymbol{o} &= \phi_{\texttt{sigmoid}}(\boldsymbol{U}^{(o)}\boldsymbol{h}^{(t-1)} + \boldsymbol{W}^{(o)}\boldsymbol{x}^{(t)} + \boldsymbol{b}^{(o)}) \\
\tilde{\boldsymbol{h}} &= \phi_{\texttt{tanh}}(\boldsymbol{U}^{(\tilde{h})}\boldsymbol{h}^{(t-1)} + \boldsymbol{W}^{(\tilde{h})}\boldsymbol{x}^{(t)} + \boldsymbol{b}^{(\tilde{h})})
\end{aligned} \tag{2.8}$$

A new update state $\tilde{\boldsymbol{h}}$ is computed in the same way as in a standard RNN. Given the gates $\boldsymbol{i}$, $\boldsymbol{f}$, $\boldsymbol{o}$, the current cell state will be the sum of the element-wise product of the forget gate of the previous cell-state and the input gate and the current update hidden state:

$$\boldsymbol{c}^{(t)} = \boldsymbol{c}^{(t-1)} \odot \boldsymbol{f} + \tilde{\boldsymbol{h}} \odot \boldsymbol{i} \tag{2.9}$$

Given the new cell state, the current output hidden state is computed via an element-wise product of the output gate and a non-linear function of the cell state:

$$\boldsymbol{h}^{(t)} = \phi_{\texttt{tanh}}(\boldsymbol{c}^{(t)}) \odot \boldsymbol{o} \tag{2.10}$$

A graphical representation of an LSTM is shown in figure 2.4. The main advantage of an LSTM over a standard RNN is that LSTMs are capable of carrying information over far longer time-spans, which allows them to capture longer-term dependencies in the data than simple RNNs. Currently LSTMs are the *default* model of RNNs for a wide range of applications, including language modelling [118], machine translation [21, 8] and generative models [97, 121]. However, newer models called *transformers* [123], which are not described in this thesis, have recently been gaining popularity.

2.1.3 Attention Mechanisms

An important component for modern NLP applications, such as Machine Translation [8], summarization [93, 139, 109] and question answering [50, 76] are *Attention Mechanisms*. An attention mechanism is an architectural component which allows the efficient compression of a variable-length sequence of vectors $\boldsymbol{h}^{(1:T)}$ into a fixed-length vector $\boldsymbol{h}$. Typically sequence-to-vector embeddings are computed either by processing via an RNN and using the final hidden-state or by averaging the sequence:

$$\boldsymbol{h} = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{h}^{(t)} \tag{2.11}$$

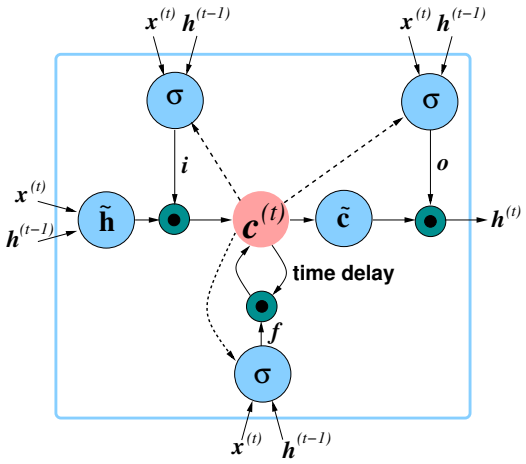


Figure 2.4 Long Short-Term Memory Network. Dotted lines represent optional peep-hole connections.

The limitation of using the final hidden state of an RNN is that typically it will preserve on only the most *recent* information in the sequence. The limitation of averaging is that all information is equally weighted. Attention mechanisms provide an approach to select only the most relevant element of the sequence $\boldsymbol{h}^{(1:T)}$ to a particular task, and discard or *de-weight* the rest. Attention mechanism are trained as part of a larger network end-to-end, where the structure of the network is such that it itself learns what is and isn't relevant to the task which the whole network is solving.

In an attention mechanism the fixed-length embedding $\boldsymbol{h}$ of the sequence $\boldsymbol{h}^{(1:T)}$ is computed as a weighted sum of the sequence $\boldsymbol{h}^{(1:T)}$ given a set of attention weights $\boldsymbol{\pi} = [\pi_1, \cdots, \pi_T]$[1] (eq. 2.12) produced by an attention mechanism.

$$\boldsymbol{h} = \sum_{t=1}^{T} \pi_t \boldsymbol{h}^{(t)}, \; \pi_t \geq 0, \sum_{t=1}^{T} \pi_t = 1 \tag{2.12}$$

The aim of the attention mechanism is to focus only on the properties of the sequence which are in some sense 'relevant' to the key vector $\boldsymbol{k}$. The attention weights for each hidden state are computed as a softmax (eq. 2.13), where the logits are given by a similarity function $s(\boldsymbol{k}, \boldsymbol{h}^{(\tau)})$ between the key vector and the hidden state.

$$\pi_t = \frac{e^{s(\boldsymbol{k}, \boldsymbol{h}^{(t)})}}{\sum_{\tau=1}^{T} e^{s(\boldsymbol{k}, \boldsymbol{h}^{(\tau)})}} \tag{2.13}$$

[1] We use the same notation for attention as for discrete probabilities because they are both non-negative and sum to one.

The similarity function (eq. 8.24) computes how strongly a hidden state relates to the key vector:

$$s(\boldsymbol{k}, \boldsymbol{h}^{(t)}) = \boldsymbol{v}^{\mathsf{T}} \tanh(\boldsymbol{\Lambda}_1 \boldsymbol{k} + \boldsymbol{\Lambda}_2 \boldsymbol{h}^{(t)} + \boldsymbol{b}) \tag{2.14}$$

where the parameters of the attention mechanism are $\{\boldsymbol{v}, \boldsymbol{\Lambda}_1, \boldsymbol{\Lambda}_2, \boldsymbol{b}\}$. This similarity function was used in [8] for neural machine translation. Alternative attention mechanisms, with different similarity functions [77] and attention sharpening [43] could potentially be used, but are not explored in this thesis.

2.1.4 Parameterizing Distributions using Neural Networks

In many machine learning tasks neural networks are used to parameterize probability distributions over either categorical random variables y or continuous random variables $\boldsymbol{y}$. Neural networks are able to do this by yielding the parameters of these distributions. For example, a neural network can parameterize a distribution over discrete classes $\mathtt{P}(y|\boldsymbol{x};\boldsymbol{\theta})$ by yielding the parameters of a categorical distribution:

$$\begin{aligned} \mathtt{P}(y|\boldsymbol{x};\boldsymbol{\theta}) &= \mathtt{Cat}(y|\hat{\boldsymbol{\pi}}) \\ \hat{\boldsymbol{\pi}} &= \boldsymbol{f}(\boldsymbol{x};\boldsymbol{\theta}), \pi_c \geq 0, \sum_{c=1}^{K} \pi_c = 1 \end{aligned} \tag{2.15}$$

Here $\boldsymbol{\pi}$ is a vector of class probabilities:

$$\boldsymbol{\pi} = \begin{bmatrix} \mathtt{P}(y = \omega_1) \\ \vdots \\ \mathtt{P}(y = \omega_K) \end{bmatrix} \tag{2.16}$$

A neural network can yield a vector of positive values which sums to one via the *softmax* output function:

$$\boldsymbol{\phi}_{\mathtt{softmax}}(\boldsymbol{x})_c = \frac{e^{x_c}}{\sum_{j=1}^{K} e^{x_j}} \;=\; \mathtt{P}(y = \omega_c|\boldsymbol{x}) = \;\hat{\pi}_c \tag{2.17}$$

Similarly, a neural network can parameterize a multivariate normal distribution over continuous targets $\boldsymbol{y} \in \mathcal{R}^K$ by predicting the mean $\hat{\boldsymbol{\mu}}$ and covariance matrix $\hat{\boldsymbol{\Sigma}}$:

$$\begin{aligned} \mathtt{p}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta}) &= \mathcal{N}(y|\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}) \\ \left\{\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}}\right\} &= \boldsymbol{f}(\boldsymbol{x};\boldsymbol{\theta}) \end{aligned} \tag{2.18}$$

Here, the mean is simply a vector of real values which can be predicted by using a linear output layer. Typically diagonal covariance matrices are considered in such models. Thus, the model can either yield the log-variances using a linear layer, or the variances using an exponential output function, for example.

2.2 Training

Having discussed the main types of neural networks and detailed standard architectural components, it is now necessary to discuss how to train neural networks for classification and regression tasks using an empirical risk minimization approach called *Maximum Likelihood Estimation*. Maximum likelihood estimation is the standard approach for training models, as it has certain desirable properties, such as consistency and statistical efficiency [40, 92], under certain conditions. The following discussions for both classification (section 2.2.1) and regression (section 2.2.2) assume the scenario where a network is trained on a finite *training data set* $\mathcal{D}_{train}$ and then evaluate on an heldout *test data set* $\mathcal{D}_{test}$. The goal of training is to make sure that the network will learn to generalize well from the training data to have good performance on the test data. Situations where the network has low empirical risk on the training data but high empirical risk on the test data is called *over-fitting*, while situations where the network has high empirical risk on both the training and test data is called *under-fitting*.

2.2.1 Training models for Classification

Consider a finite dataset $\mathcal{D}_{train} = \{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$ sampled from an underlying data distribution $\mathtt{p_{tr}}(\boldsymbol{x}, y)$ where $\boldsymbol{x} \in \mathcal{R}^D$ and $y \in \{\omega_1, \cdots, \omega_K\}$. The goal is to train a discriminative classification model $\mathtt{P}(y|\boldsymbol{x}; \hat{\boldsymbol{\theta}})$[2] parameterized via a neural network:

$$\begin{aligned} \mathtt{P}(y|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}}) &= \mathtt{Cat}(y; \hat{\boldsymbol{\pi}}) \\ \hat{\boldsymbol{\pi}} &= \boldsymbol{f}(\boldsymbol{x}^*; \hat{\boldsymbol{\theta}}),\ \sum_{c=1}^{K} \hat{\pi}_c = 1,\ \pi_c \geq 0 \end{aligned} \tag{2.19}$$

Typically this is a feed-forward network with a softmax output function which produces a vector of categorical probabilities $\hat{\boldsymbol{\pi}}$. The model $\mathtt{P}(y|\boldsymbol{x}; \boldsymbol{\theta})$ is trained via *Maximum Likelihood*

[2] $\mathtt{P}(\omega_c|\boldsymbol{x}; \hat{\boldsymbol{\theta}})$ is a shorthand for $\mathtt{P}(y = \omega_c|\boldsymbol{x}; \hat{\boldsymbol{\theta}})$

Estimation to maximize the likelihood of the observed data:

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \arg\max_{\boldsymbol{\theta}}\{\mathtt{P}(Y|\boldsymbol{X};\boldsymbol{\theta})\} \\ &= \arg\max_{\boldsymbol{\theta}}\{\prod_{i=1}^{N}\mathtt{P}(y^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta})\} \\ &= \arg\max_{\boldsymbol{\theta}}\{\prod_{i=1}^{N}\prod_{c=1}^{K}\mathtt{P}(\hat{y}=\omega_c|\boldsymbol{x}^{(i)};\boldsymbol{\theta})^{\mathcal{I}(y^{(i)}=\omega_c)}\}\end{aligned} \tag{2.20}$$

where $\mathcal{I}$ is an indicator function. Directly optimizing this expression is inconvenient, as the product of probabilities can yield numerically unstable behaviour in floating-point numbers. A far more stable loss to maximize is *log-likelihood*, which yields the same maximum, as log is a monotonic function of the input. This transforms the loss from a product of probabilities into a sum of log-probabilities. Furthermore, as it is customary to *minimize* a loss function, instead of *maximizing* the *log-likelihood* we *minimize* the *negative log-likelihood*. Finally, this loss can be expressed as an expectation by diving by N, which does not affect the location of the maximum, yielding:

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \arg\min_{\boldsymbol{\theta}}\big\{-\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{K}\mathcal{I}(y^{(i)}=\omega_c)\ln\mathtt{P}(\hat{y}=\omega_c|\boldsymbol{x}^{(i)};\boldsymbol{\theta})\big\} \\ &= \arg\min_{\boldsymbol{\theta}}\big\{\mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x},y)}\big[\mathcal{L}^{NLL}(y,\boldsymbol{x},\boldsymbol{\theta})\big]\big\}\end{aligned} \tag{2.21}$$

where the expectation is taken with respect to the *empirical distribution* $\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x},y)=\mathcal{D}_{train}$. This yields the standard form of *negative log-likelihood loss* used for training classification models. This is a form of empirical risk minimization, where the risk is the negative log-likelihood.

$$\mathcal{L}^{NLL}(y,\boldsymbol{x},\boldsymbol{\theta}) = -\sum_{c=1}^{K}\mathcal{I}(y=\omega_c)\ln\mathtt{P}(\hat{y}=\omega_c|\boldsymbol{x};\boldsymbol{\theta}) \tag{2.22}$$

In this thesis the notation $\mathcal{L}(\boldsymbol{\theta},\mathcal{D})$ will be a short hand notation for the expectation of the loss with respect to the *true* underlying distribution $\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x},y)$:

$$\mathcal{L}(\boldsymbol{\theta},\mathcal{D})=\mathbb{E}_{\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x},y)}[\mathcal{L}^{NLL}(y,\boldsymbol{x},\boldsymbol{\theta})] \tag{2.23}$$

A similar notation will be a shorthand for the expectation with respect to a specific finite dataset $\mathcal{L}(\boldsymbol{\theta},\mathcal{D}_{train})$:

$$\mathcal{L}(\boldsymbol{\theta},\mathcal{D}_{train})=\mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x},y)}[\mathcal{L}^{NLL}(y,\boldsymbol{x},\boldsymbol{\theta})] \tag{2.24}$$

Further insight can be obtained by taking the expectation of this loss with respect to the *true* underlying distribution $\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x}, y)$:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}, \mathcal{D}) &= \mathbb{E}_{\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x})}\Big[-\sum_{c=1}^{K} \mathbb{E}_{\mathtt{P}_{\mathtt{tr}}(y|\boldsymbol{x})}[\mathcal{I}(y=\omega_c)] \ln \mathtt{P}(\hat{y}=\omega_c|\boldsymbol{x};\boldsymbol{\theta})\Big] \\
&= \mathbb{E}_{\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x})}\Big[-\sum_{c=1}^{K} \mathtt{P}_{\mathtt{tr}}(\omega_c|\boldsymbol{x}) \ln \mathtt{P}(\omega_c|\boldsymbol{x};\boldsymbol{\theta})\Big] \\
&= \mathbb{E}_{\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x})}\Big[\sum_{c=1}^{K} \mathtt{P}_{\mathtt{tr}}(\omega_c|\boldsymbol{x}) \ln \frac{\mathtt{P}_{\mathtt{tr}}(\omega_c|\boldsymbol{x})}{\mathtt{P}(\omega_c|\boldsymbol{x};\boldsymbol{\theta})} - \mathtt{P}_{\mathtt{tr}}(\omega_c|\boldsymbol{x}) \ln \mathtt{P}_{\mathtt{tr}}(\omega_c|\boldsymbol{x})\Big] \\
&= \mathbb{E}_{\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x})}\Big[\underbrace{\mathtt{KL}[\mathtt{P}_{\mathtt{tr}}(y|\boldsymbol{x})||\mathtt{P}(y|\boldsymbol{x};\boldsymbol{\theta})]}_{Reducible\ Loss} + \underbrace{\mathcal{H}[\mathtt{P}_{\mathtt{tr}}(y|\boldsymbol{x})]}_{Irreducible\ Loss} \Big]
\end{aligned} \tag{2.25}$$

This derivation shows that maximizing the likelihood of the data is equivalent to minimizing the KL-divergence between the model and the underlying data distribution, where the loss is lower bounded by the expected entropy of the conditional distribution $\mathcal{H}[\mathtt{P}_{\mathtt{tr}}(y|\boldsymbol{x})]$. However, it is necessary to point out that this derivation is only valid when there is an infinite amount of training data and when the true distribution $\mathtt{P}_{\mathtt{tr}}(y|\boldsymbol{x})]$ is within the family of models which can be captured by model being trained $\mathtt{P}(y|\boldsymbol{x};\boldsymbol{\theta})$. Consider a Monte-Carlo approximation to the expectation for a finite dataset:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}, \mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N}\Big[\sum_{c=1}^{K}\frac{1}{N_i}\sum_{j=1}^{N_i} -\mathcal{I}(y^{(ij)}=\omega_c)\ln \mathtt{P}(\hat{y}=\omega_c|\boldsymbol{x}^{(i)};\boldsymbol{\theta})\Big] \\
\boldsymbol{x}^{(i)} \sim \mathtt{p}_{\mathtt{tr}}(\boldsymbol{x}),\ y^{(ij)} \sim \mathtt{P}_{\mathtt{tr}}(y|\boldsymbol{x}^{(i)})
\end{aligned} \tag{2.26}$$

For the derivation to be accurate, the number N_i of samples $y^{(ij)} \sim \mathtt{P}_{\mathtt{tr}}(y|\boldsymbol{x}^{(i)})$ must be large in order to appropriately capture the conditional distribution at a particular $\boldsymbol{x}^{(i)}$. In practice, however, $N_i = 1$ and only a single sample of $y^{(ij)}$ of obtained for every $\boldsymbol{x}^{(i)}$. In the case that the data exists on a sufficiently smooth manifold this may not be a problem given that the inputs $\boldsymbol{x}$ cover the manifold with sufficient density. This implies that the number N of samples of $\boldsymbol{x}^{(i)}$ has to be large. Consequently, a model trained on a *finite* dataset sampled from the underlying distribution may only model part of the space and poorly generalize to a different *test* dataset sampled from the same underlying distribution $\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x}, y)$.

2.2.2 Training models for Regression

Maximum likelihood training of regression models is similar to MLE training of classification, the primary difference is that the targets are points in a continuous space rather than discrete class labels. Consider a finite dataset $\mathcal{D}_{train} = \{\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\}_{i=1}^{N}$ sampled from an underlying data distribution $\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x}, \boldsymbol{y})$ where $\boldsymbol{x} \in \mathcal{R}^D$ and $\boldsymbol{y} \in \mathcal{R}^K$. The goal is to train a *Density Network* $\mathtt{p}(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta})$ to maximize the likelihood of the training data as follows:

$$\begin{aligned}
\hat{\boldsymbol{\theta}} &= \arg\max_{\boldsymbol{\theta}} \big\{ \mathtt{p}(\boldsymbol{Y}|\boldsymbol{X}; \boldsymbol{\theta}) \big\} \\
&= \arg\max_{\boldsymbol{\theta}} \big\{ \prod_{i=1}^{N} \mathtt{p}(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) \big\} \\
&= \arg\min_{\boldsymbol{\theta}} \big\{ \frac{1}{N} \sum_{i=1}^{N} -\ln \mathtt{p}(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) \big\} \\
&= \arg\min_{\boldsymbol{\theta}} \big\{ \mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x}, \boldsymbol{y})} \big[\mathcal{L}^{NLL}(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}) \big] \big\}
\end{aligned} \tag{2.27}$$

As in the derivation for classification, maximum-likelihood estimation is done via *minimization* of the expectation of the *negative log-likelihood* with respect to the empirical distribution $\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x}, \boldsymbol{y}) = \mathcal{D}_{train}$. Similar to the derivation 2.25 it is possible to show that in expectation minimizing negative log-likelihood is equivalent to minimizing the expected KL-divergence between the model and the underlying data distributions:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}, \mathcal{D}) &= \mathbb{E}_{\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x})} \Big[\mathbb{E}_{\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x})} [-\ln \mathtt{p}(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta})] \Big] \\
&= \mathbb{E}_{\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x})} \Big[-\int \mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x}) \ln \mathtt{p}(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta}) d\boldsymbol{y} \Big] \\
&= \mathbb{E}_{\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x})} \Big[\int \mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x}) \ln \frac{\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x})}{\mathtt{p}(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta})} - \mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x}) \ln \mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x}) d\boldsymbol{y} \Big] \\
&= \mathbb{E}_{\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x})} \Big[\underbrace{\mathtt{KL}[\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x}) || \mathtt{p}(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta})]}_{Reducible\ Loss} + \underbrace{\mathcal{H}[\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x})]}_{Irreducible\ Loss} \Big]
\end{aligned} \tag{2.28}$$

This derivation, just like in the case for classification, requires that a large amount of data is available to train the model and that the true distribution falls within the family of models $\mathtt{p}(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta})$. A crucial difference from the derivation 2.25 is that it is also necessary to assume the correct output density in order to be able to fully minimize this loss. Specifically, while all conditional distributions over discrete classes can be parameterized by a softmax output, there is a large variety of possible continuous distributions over the continuous targets $\boldsymbol{y}$, each reflecting a different set of assumptions about $\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x})$. A common choice of output

distribution is the multivariate normal (MVN) distribution with a diagonal covariance matrix[3]. In this case, the neural network will yield the mean and variances, conditioned on the input $\boldsymbol{x}$:

$$\begin{aligned} \mathtt{p}(\boldsymbol{y}|\boldsymbol{x};\hat{\boldsymbol{\theta}}) &= \mathcal{N}(\boldsymbol{y};\hat{\boldsymbol{\mu}},\hat{\boldsymbol{\Sigma}}) \\ \{\hat{\boldsymbol{\mu}},\hat{\boldsymbol{\Sigma}}\} &= \boldsymbol{f}(\boldsymbol{x};\hat{\boldsymbol{\theta}}) \end{aligned} \tag{2.29}$$

Alternative choices of output distribution are possible, including, but not limited to, the Laplace distribution, the multivariate Student's T distribution or a Generalized Normal distribution.

Unfortunately, the true distribution $\mathtt{p}(\boldsymbol{y}_{tr}|\boldsymbol{x})$ may be multi-modal and non-symmetric, making common choices of output distribution inappropriate. In these cases it is possible to parameterize a mixture of normal distributions using a *Mixture Density Network* [10]:

$$\begin{aligned} \mathtt{p}(\boldsymbol{y}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}}) &= \sum_{q=1}^{Q} \mathcal{N}(\boldsymbol{y};\hat{\boldsymbol{\mu}}^{(q)},\hat{\boldsymbol{\Sigma}}^{(q)})\hat{\pi}_q \\ \{\hat{\boldsymbol{\mu}}^{(1:Q)},\hat{\boldsymbol{\Sigma}}^{(1:Q)},\hat{\boldsymbol{\pi}}\} &= \boldsymbol{f}(\boldsymbol{x}^*;\hat{\boldsymbol{\theta}}),\ 0 \leq \pi_q,\ \sum_{q=1}^{Q}\pi_q = 1 \end{aligned} \tag{2.30}$$

where $\hat{\boldsymbol{\mu}}^{(q)}$ and $\hat{\boldsymbol{\Sigma}}^{(q)}$ are the mean and variance of the q-th mixture component. The mixture weights $\hat{\boldsymbol{\pi}}$ are produced via a softmax output head. A mixture of normal distributions can approximate an arbitrary distribution $\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x})$ given a sufficient number of mixture components.

It is necessary to point out that the reason we are interested in accurately capturing the distribution $\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x})$, rather than just a point estimate $\hat{\boldsymbol{y}}$, is because we are interested probabilistic models which can yield uncertainty estimates. Models which only yield a point estimates, called *Regressors*, are more commonly used for regression tasks than Density Networks. However, it can be shown that Regressors, which are commonly trained with an L_2 loss, are a special case of a Density Network. Consider the negative log-likelihood for a model which parameterizes a multivariate normal output density with a fixed covariance

[3]It is possible to use a full-covariance matrix to model variable correlation, but is not commonly done

$\hat{\boldsymbol{\Sigma}} = \boldsymbol{I}$:

$$\begin{aligned}
\mathcal{L}^{NLL}(\boldsymbol{y}^{(i)}, \boldsymbol{x}^{(i)}, \boldsymbol{\theta})) &= -\ln \mathrm{p}(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)}; \boldsymbol{\theta}) \\
&= -\ln\left(\frac{1}{\sqrt{|2\pi\hat{\boldsymbol{\Sigma}}|}} e^{-\frac{1}{2}(\boldsymbol{y}^{(i)}-\hat{\boldsymbol{\mu}}^{(i)})^T\hat{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{y}^{(i)}-\hat{\boldsymbol{\mu}}^{(i)})}\right) \\
&= \frac{1}{2}(\boldsymbol{y}^{(i)} - \hat{\boldsymbol{\mu}}^{(i)})^T\hat{\boldsymbol{\Sigma}}^{-1}(\boldsymbol{y}^{(i)} - \hat{\boldsymbol{\mu}}^{(i)}) + \frac{1}{2}\ln(|2\pi\hat{\boldsymbol{\Sigma}}|) \\
&= \frac{1}{2}||(\boldsymbol{y}^{(i)} - \hat{\boldsymbol{\mu}}^{(i)})||_2^2 + \frac{K}{2}\ln(2\pi), \iff \hat{\boldsymbol{\Sigma}} = \boldsymbol{I}
\end{aligned} \tag{2.31}$$

Thus, minimization of negative log-likelihood for a multivariate normal density network with a fixed covariance equal to the identity matrix is equivalent to minimization of the square of the L_2 norm between the network output and the target. Similarly, it can be shown other L_p losses are special cases of Maximum Likelihood Estimation with different output densities and a fixed output variance. Throughout the rest of this thesis only density networks will be considered for regression tasks.

2.2.3 Regularization

As discussed in sections 2.2.1 and 2.2.2, derivations 2.25 and 2.28 only hold in the infinite data scenario. In practice, it is necessary to operate on finite datasets. In order to improve the generalization to a previously heldout test dataset $\mathcal{D}_{test}$ different forms of *regularization* are used. The four most common forms of regularization for deep learning are *weight decay*, *early stopping*, *data augmentation* and *dropout* [11, 92, 40, 117]. There are many other forms of regularization, but their discussion is beyond the scope of this thesis.

Weight decay is one of the oldest known forms of regularization which puts an L_2 norm penalty on each of the weights and prevents them from growing too large, thereby preventing over-fitting. This has additional benefits, such as stabilizing training and smoothing the function learned by the neural network:

$$\mathcal{L}(\boldsymbol{\theta}, \mathcal{D}_{train}) = \mathcal{L}^{NLL}(\boldsymbol{\theta}, \mathcal{D}_{train}) + \frac{\lambda}{2}||\boldsymbol{\theta}||_2^2 \tag{2.32}$$

Weight decay can be seen as a form of Maximum A-Posteriori (MAP) estimation where a zero-mean normal prior $\mathrm{p}(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\mathbf{0}, \sigma_p^2 \cdot \boldsymbol{I})$ with a variance of σ_p^2 is assumed over the parameters $\boldsymbol{\theta}$:

$$\begin{aligned}
\hat{\boldsymbol{\theta}} &= \arg\max_{\boldsymbol{\theta}}\{\mathrm{P}(\boldsymbol{y}|\boldsymbol{X}; \boldsymbol{\theta})\mathrm{p}(\boldsymbol{\theta})\} \\
&= \arg\min_{\boldsymbol{\theta}}\left\{\mathcal{L}^{NLL}(\boldsymbol{\theta}, \mathcal{D}_{train}) - \ln \mathrm{p}(\boldsymbol{\theta})\right\}
\end{aligned} \tag{2.33}$$

Using other L_p norms is equivalent to assuming a different prior distribution $\mathrm{p}(\boldsymbol{\theta})$ over the weights [40, 11, 92].

Another well known form of regularization is called *early stopping* [40, 11, 92]. This form of regularization requires three datasets - $\mathcal{D}_{train}$, $\mathcal{D}_{valid}$ and $\mathcal{D}_{test}$, all assumed to be sampled from the same underlying distribution. The model is trained to minimize the expected negative log-likelihood of the training data, while at the same time keeping track of the expected negative log-likelihood on the validation dataset. The loss on the validation dataset is considered to be a proxy for the loss of the model on a heldout test dataset, which also comes from the same underlying distribution. When the loss on the validation set stops decreasing and starts rising, training of the model is stopped, thereby preventing the model from over-fitting to the training data.

The best way to improve model generalization and decrease over-fitting is to simply use more data. Unfortunately, it is not always possible to gather additional training data, especially for supervised tasks. However, it is possible *augment* an existing training dataset with a set of valid perturbations of the training data [40]. For example, images of objects can be flipped, rotated and shifted, as these transformations do not alter the 'content' or class of the image while adding new training data points which teach the model to be robust to realistic perturbations of the data. However, care must to taken to only augment using valid transformations. For example, in a digit or character classification task it is not valid to flip the digits up/down or left/right, as that will alter the content, but shifts and minor rotations are acceptable. Unfortunately, in domains other than image classification, it may be difficult to define a set of transformations, which limits the use of data augmentation to mainly image-related tasks.

Finally, a popular type of regularization is called *Dropout* [117, 116, 56], which encourages neural networks to learn robust representations. Dropout regularization works in two phases: training and inference. During training, for every input, the activations of every hidden layer are randomly zeroed-out by multiplying by a binary mask-vector sampled from a Bernoulli distribution with a probability of producing 1 equal to p:

$$\boldsymbol{h}^{(l)} = \frac{1}{p}\boldsymbol{b} \odot \boldsymbol{h}^{(l)},\ \boldsymbol{b} \sim \mathcal{B}(p) \tag{2.34}$$

The resulting activations are multiplied by $\frac{1}{p}$ in order to maintain the same expected activation magnitude as when dropout is not used. During inference no Bernoulli noise is used and all units in a hidden layer are active. By randomly zeroing out hidden units dropout regularization encourages the network to learn intermediate feature representations which are informative and robust *by themselves*, rather than only in very specific combinations. Furthermore, this

can be seen as a form of model-averaging. By training multiple *sub-nets* within the same network to learn the same function, and using the *mean network* during inference dropout can be seen as a form of *pseudo-ensemble*. It is possible to use an explicit form of model averaging via *Monte-Carlo Dropout*, where for the same test input $\boldsymbol{x}^*$ different dropout masks are used, and the predictions given each dropout mask are averaged together:

$$\mathtt{P}(y|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}}) = \frac{1}{M}\sum_{m=1}^{M}\mathtt{P}(y|\boldsymbol{x}^*,\boldsymbol{b}^{(m)};\hat{\boldsymbol{\theta}}),\ \boldsymbol{b}^{(m)} \sim \mathcal{B}(p) \tag{2.35}$$

This can sometimes yield a performance benefit at increased computational cost, as it requires re-running the networks for the same input M times.

2.3 Optimization

Having discussed objectives to optimize in section 2.2, it is now necessary to discuss how to actually carry out this optimization with respect to the model parameters $\boldsymbol{\theta}$. Unfortunately, it is not possible to obtain closed-form solutions for the optimization problem defined in the previous section in equations 2.20 and 2.27 for deep neural networks. Instead, neural networks are optimized using iterative gradient descent methods [107]. Currently, there exists a wide range of gradient descent optimizers for neural networks, including stochastic gradient descent with momentum [11, 40], RMSProp [40], Adagrad [29], AdaDelta [138] and Adam [62]. The current section describes gradient descent optimization, detailing the Adam optimizer [62] used to train all models throughout this thesis, and discusses the topics of learning rate scheduling and initialization.

2.3.1 Gradient Descent Optimization

Gradient descent methods can be broadly split into *Batch* gradient descent and *Stochastic* gradient descent. Batch Gradient Descent minimizes the loss over the entire training dataset per-iteration t. A full pass over the entire training dataset is called an *epoch*, and for batch gradient descent each iteration is one epoch. The model is trained by updating the parameters with the gradient of a loss function $\mathcal{L}(y^{(i)}, \boldsymbol{x}^{(i)}, \boldsymbol{\theta})$ with respect to the parameters $\boldsymbol{\theta}$, multiplied by the learning rate η: Where the gradient of the loss with respect to each parameter is obtained via the *Error Back-propagation* (BP) [107] and *Error Back-Propagation Through Time* (BPTT) [87, 88, 86] algorithms for feed-forward and recurrent networks, respectively. Batch Gradient Descent yields the exact gradient of the loss on the training data, but may take a long time to iterate over datasets of millions or billions of data points.

Algorithm 1 Batch Gradient Descent

while $t \leq T$ **do**
$\boldsymbol{\theta}^{(t+1)} \leftarrow \boldsymbol{\theta}^{(t)} - \eta \cdot \frac{1}{N} \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} \mathcal{L}^{NLL}(y^{(i)}, \boldsymbol{x}^{(i)}, \boldsymbol{\theta})$
end while

Instead of doing a full pass over the dataset before updating the parameters, it is possible to update the parameters after processing each datapoint, doing N parameter updates per epoch. This is called *Stochastic Gradient Descent* [105]. This approach yields significantly

Algorithm 2 Stochastic Gradient Descent

while $t \leq T \cdot N$ **do**
$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \cdot \nabla_{\boldsymbol{\theta}} \mathcal{L}^{NLL}(y^{(t)}, \boldsymbol{x}^{(t)}, \boldsymbol{\theta})$
end while

faster convergence than Batch Gradient Descent, but at the cost of noisy gradient updates. In Stochastic Gradient Descent the gradient of the loss at each datapoint is a single-sample approximation to the gradient of the loss over the entire dataset. As a consequence, it may be necessary to use a far lower learning rate to avoid instabilities during training.

A compromise between Batch and Stochastic Gradient Descent is *Stochastic Minibatch Gradient Descent*, which minimizes the loss on stochastic *minibatches* of size $N_b << N$. The size of the minibatch is typically between 16 and 256 datapoints. Sizes are given in powers of 2 because of the way data is best partitioned on modern GPUs. However, seeing the same training examples in the same order every epoch may introduces unnecessary biases into the model. Thus, mini-batches are shuffled between each epoch via uniform sampling from the training data without replacement. In practice, Stochastic Gradient Descent and minibatch shuffling can be seen as forms of regularization, as the noise added to the gradients may prevent the model from over-fitting to the training dataset and avoid getting stuck in a local minimum.

Algorithm 3 Stochastic Minibatch Gradient Descent

while $t \leq T \cdot \frac{N}{N_b}$ **do**
$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \cdot \frac{1}{N_b} \sum_{i=1}^{N_b} \nabla_{\boldsymbol{\theta}} \mathcal{L}^{NLL}(y^{(i)}, \boldsymbol{x}^{(i)}, \boldsymbol{\theta}), \quad \{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^{N_b} \sim \mathcal{D}_{train}$
end while

If stochastic gradient descent yields particularly noisy gradients or the curvature of the loss surface is strong in some directions and weak in others, gradient descent may yield 'zig-zagging' behaviour along the loss surface which slows convergence. *Momentum methods* are typically used to avoid this behaviour and accelerate convergence by adding the previous

parameter update, multiplied by the momentum rate α where $0 \leq \alpha \leq 1$, to the current parameter update. This increases the effective learning rate in the direction of consistent gradients and accelerates convergence.

Algorithm 4 Stochastic Minibatch Gradient Descent with Momentum

while $t \leq T \cdot \frac{N}{N_b}$ **do**
$\boldsymbol{m}^{(t)} = (1-\alpha)\cdot\big(\frac{1}{N_b}\sum_{i=1}^{N_b} \nabla_{\boldsymbol{\theta}} \mathcal{L}^{NLL}(y^{(i)}, \boldsymbol{x}^{(i)}, \boldsymbol{\theta})\big) + \alpha \cdot \boldsymbol{m}^{(t-1)}, \ \{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^{N_b} \sim \mathcal{D}_{train}$
$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \cdot \boldsymbol{m}^{(t)}$
end while

Currently, *Adam* [62] is a popular state-of-the-art stochastic gradient based optimization algorithm for neural networks. Adam, as well as other adaptive learning rate methods like Adagrad [29] and RMSprop [40] are pseudo-second order methods which attempt to account for the curvature of the loss surface along each dimension. The advantage of Adam is that it combines momentum with automatic adjustment of the learning rate for each parameter.

Algorithm 5 Adam

$\boldsymbol{s}^{(0)} = \boldsymbol{r}^{(0)} = \mathbf{0}$
while $t \leq T \cdot \frac{N}{N_b}$ **do**
$\boldsymbol{g}^{(t)} = \frac{1}{N_b}\sum_{i=1}^{N_b} \nabla_{\boldsymbol{\theta}} \mathcal{L}^{NLL}(y^{(i)}, \boldsymbol{x}^{(i)}, \boldsymbol{\theta}), \quad \{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^{N_b} \sim \mathcal{D}_{train}$
$\boldsymbol{s}^{(t)} = \beta_1 \cdot \boldsymbol{s}^{(t-1)} + (1-\beta_1) \cdot \boldsymbol{g}^{(t)}$
$\boldsymbol{r}^{(t)} = \beta_2 \cdot \boldsymbol{r}^{(t-1)} + (1-\beta_2) \cdot \boldsymbol{g}^{(t)} \odot \boldsymbol{g}^{(t)}$
$\hat{\boldsymbol{s}}^{(t)} = \frac{\boldsymbol{s}^{(t)}}{1-\beta_1^t}$
$\hat{\boldsymbol{r}}^{(t)} = \frac{\boldsymbol{r}^{(t)}}{1-\beta_2^t}$
$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta \frac{\hat{\boldsymbol{s}}^{(t)}}{\sqrt{\hat{\boldsymbol{r}}^{(t)}}+\epsilon}$
end while

In Adam, the biased estimates of the first and second moments of the gradients, $\boldsymbol{s}^{(t)}$ and $\boldsymbol{r}^{(t)}$, are computed at each iteration. The biased estimate of the first moment $\boldsymbol{s}^{(t)}$ directly incorporates momentum into the algorithm. The first and second moments of the gradients are then de-biased, yielding the unbiased estimates of the first and second moments. This de-biasing is important only at the beginning of training and as training progresses the unbiased estimates tend to the biased estimates. The parameters are then updated by subtracting the ratio of the unbiased estimate of the first moment $\hat{\boldsymbol{s}}^{(t)}$ and the square root of the unbiased estimate of the second moment $\hat{\boldsymbol{r}}^{(t)}$, multiplied by the global learning rate η. This process is controlled by the hyper parameters β_1, β_2 and ϵ, whose typical values are 0.9, 0.999 and 1e-8, respectively.

The potential drawbacks of Adam are that the estimates of the second order moments may become stale near a local minimum and that Adam does not properly interact with weight decay regularization. Several extensions to Adam have been proposed to address these issues, such as Adamax[62], AMSGrad [104], and AdamW [75], but the performance improvements are not consistent. Thus, throughout this thesis the standard version of Adam is used to train all neural network models.

2.3.2 Learning Rate Schedules

The learning rate η plays an important role in gradient descent optimization. If a learning rate is too high, then training may become unstable, and if the learning rate is too low, then it may take the optimization process a long time to converge to a local minimum. Furthermore, due to the noisy nature of the gradients, stochastic gradient descent and minibatch stochastic gradient descent will not converge to a local minimum without a decaying learning rate. Thus, it is necessary to decay the learning rate over the course of training, so that the optimization can settle in some local (or global) minimum. Typically, an exponentially decaying learning rate is used:

$$\eta^{(t)} = \eta^{(0)} e^{-\frac{t}{\lambda}} \tag{2.36}$$

where λ is the decay constant and η_0 is the initial learning rate.

It is possible to consider alternative learning rate schedules. Currently cyclical learning rates are popular, as they allow for faster optimization. In particular, in this work the *1-Cycle Policy* [115, 114] is sometimes used instead of the exponentially decaying learning rates. Here, a cycle length is defined in terms of a number of epochs. The learning rate is linearly increased from $\eta^{(0)}$ to $10 * \eta^{(0)}$ for half the cycle and then linearly decayed back down to $\eta^{(0)}$ for the second half of the cycle. Then the learning rate is linearly decay to $\frac{\eta^{(0)}}{100}$ for the remaining number of epochs. Fast Ai [45] report that this approach allows for significantly faster training of neural networks, which is why is was adopted in this work.

2.3.3 Initialization

An important factor in the success or failure of training a neural network is the choice of initialization scheme for the parameters $\boldsymbol{\theta}$. Initialization is known to have a strong effect on the optimization process and the generalization of the resulting model, though how exactly initialization affects the latter property is not fully understood [40]. Furthermore, the effects of initialization on the network's capacity to generalize are more pronounced when there is

less training data. As the quantity of available training data increase initialization effects play a smaller role in generalization.

In general, initialization must accomplish two tasks - it must induce *symmetry breaking* and avoid saturating the non-linear activations, so that gradients can propagate effectively throughout the entire network. Symmetry breaking is necessary to force each neuron in each hidden layer to learn a *different* function, otherwise the effective capacity of the network is diminished. This can be accomplished by randomly initializing the parameters. However, the choice of distribution from which the parameters are initialized can have a strong affect on the optimization process. If the variance of the distribution is too large, then the non-linear activations may saturate or explode and impede gradient flow. On the other hand, if the variance is too small, then information may poorly flow through the network. Work by Xavier and Bengio [38] suggests initializing from either a uniform distribution or a normal distribution, where the bounds on the uniform distribution or the variance of the normal distribution is a function of the number of neurons in the previous layer ($\texttt{fan}_{\texttt{in}}$) and the number of neurons in the following layer ($\texttt{fan}_{\texttt{out}}$):

$$\begin{aligned} \theta &\sim \mathcal{U}(-\sqrt{\frac{6}{\texttt{fan}_{\texttt{in}} + \texttt{fan}_{\texttt{out}}}}, \sqrt{\frac{6}{\texttt{fan}_{\texttt{in}} + \texttt{fan}_{\texttt{out}}}} \\ \theta &\sim \mathcal{N}(0, \sqrt{\frac{2}{\texttt{fan}_{\texttt{in}} + \texttt{fan}_{\texttt{out}}}}) \end{aligned} \tag{2.37}$$

The motivation is to make sure that the scale of the activation of every neuron is such that the activation function is within a linear region and does not saturate, which enables good gradient flow throughout the entire network. This is especially important for very deep networks. From empirical evidence, the particular choice of uniform or normal distribution does not seem matter greatly.

2.4 Chapter Summary

This chapter introduced the theory necessary to support discussions throughout the rest of this thesis. Section 2.1 introduced neural networks and deep learning, presenting them as powerful, trainable feature extractors which project the input features into a space where they can be linearly classified or regressed. The main architectural components of deep learning models considered in this thesis, such as fully-connected layers, convolutions layers, recurrent networks layers and attention mechanisms were discussed. Fully-connected layers are used in *each* neural network trained in this thesis. Convolution layers are used in models trained on image datasets throughout chapter 5. Bidirectional Long Short-Term Memory

(LSTM) recurrent neural networks and attention mechanisms are used to construct relevance assessment models operating over variable length prompts and responses in chapter 8.

Section 2.2 discussed the proprieties of maximum likelihood estimation for both classification and regression tasks. It was shown that a model trained via maximum likelihood will recover the true underlying distribution of data, given a sufficiently large amount of training data and correct model class. This result is particularly relevant to the estimation of *data uncertainty* discussed in the next chapter.

Section 2.3 discussed stochastic minibatch gradient descent and detailed the Adam [62] optimization algorithm, which is used to train all neural network models in this thesis. Additionally, several initialization schemes were described and discussed in the context of their effect on gradient-descent optimization.

Chapter 3

Predictive Uncertainty Estimation

The previous chapter discussed common architectural components of neural networks, such as feed-forward and recurrent layers, maximum likelihood training of parametric models for classification and regression, and optimization techniques. This chapter introduces the area of uncertainty estimation for discriminative parametric classification and regression models. The sources of uncertainty in predictions, specifically *data uncertainty*, which occurs due to noise and class overlap in the data, and *knowledge uncertainty*, which occurs when the training and test data are mismatched, are discussed in section 3.1 in the context of both classification and regression tasks. In section 3.2 it is shown that probabilistic classification and regression models will naturally capture estimates of *data uncertainty* as a consequence of maximum likelihood training, given certain conditions. Capturing estimates of *knowledge uncertainty*, however, is more difficult. In this chapter, two classes of approaches for capturing *knowledge uncertainty* are considered. Approaches based on deriving estimates of *knowledge uncertainty* from a neural network parameterized using a single set of model parameters, also know as *single model approaches*, are discussed in sections 3.3. Approaches to estimation of *knowledge uncertainty* using an ensemble of models, where each model is parameterized by a unique set of mode parameters, are discussed in section 3.4. This chapter closes with a discussion of the limitations in modelling *knowledge uncertainty* in section 3.5.

3.1 Sources of Uncertainty

In order to understand the problem of uncertainty estimation it is first necessary to understand the sources of uncertainty in predictions. Conceptually, there are two fundamental sources of uncertainty. Firstly, uncertainty in predictions can arise due to the *irreducible uncertainty* inherent in the data, which is known as *data uncertainty* or *aleatoric uncertainty*. *Data uncertainty* arises due to the complexity, multi-modality and noise in the data. *Data uncertainty*

is irreducible because it is a property of the underlying distribution which generated the data, rather than a property of the model.

The second source of uncertainty is *epistemic* or *knowledge uncertainty*, and represents uncertainty in the model's predictions due to a *lack of understanding* or *knowledge* on the part of the model regarding the current input for which the model is making a prediction. [1] In other words, this form of uncertainty arises when the *test input* $\boldsymbol{x}^*$ comes from a different distribution than the one which generated the training data. This is why the name *distributional uncertainty* is also used. Mismatch between the test and training distributions is also known as dataset shift [102] and is a situation which often arises for real world problems. As *knowledge uncertainty* uncertainty is a property of the model, unlike *data uncertainty*, it can be reduced by providing more knowledge, in the form of training data, to the model. In this thesis the names *knowledge uncertainty* or *distributional uncertainty* will be used interchangeably.

The ability to separately model the sources of uncertainty in predictions is important, as different actions can be taken by the model depending on the source of uncertainty. For example, in active learning [112] tasks collecting training data from regions with high *data uncertainty* is pointless, as that uncertainty is inherently irreducible. On the other hand, collecting additional training data from regions associated with high *knowledge uncertainty* will provide more knowledge, which should improve the quality of the model and its capacity to generalize.

3.1.1 Uncertainty for Classification

This section discusses the sources of uncertainty in the context of classification tasks, providing an illustrative example of each source of uncertainty on artificial datasets. Consider a finite dataset $\mathcal{D}_{tr}$ from a distribution $\mathtt{p_{tr}}(\boldsymbol{x}, y)$ over inputs $\boldsymbol{x} \in \mathcal{R}^D$ and class labels $y \in \{\omega_1, \cdots, \omega_K\}$:

$$\begin{aligned} \mathcal{D}_{tr} &= \{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^N \\ \{\boldsymbol{x}^{(i)}, y^{(i)}\} &\sim \mathtt{p_{tr}}(\boldsymbol{x}, y) \end{aligned} \tag{3.1}$$

[1] In general, the term *epistemic* uncertainty covers both uncertainty in predictions for in-domain test inputs $\boldsymbol{x}^*$ due to data sparsity as well as uncertainty due to distributional mismatch. However, in this thesis we use a *narrower* definition of *epistemic uncertainty*, which we refer to as *knowledge uncertainty*, which covers only the latter.

In the context of a discriminative classification task, the *data uncertainty* at an input point $\boldsymbol{x}$ is defined as the entropy of the *true* conditional distribution $\mathcal{H}[\mathtt{p}_{\mathtt{tr}}(y|\boldsymbol{x})]$:

$$\mathcal{H}[\mathtt{P}_{\mathtt{tr}}(y|\boldsymbol{x})] = -\sum_{c=1}^{K} \mathtt{P}_{\mathtt{tr}}(y=\omega_c|\boldsymbol{x}) \ln \mathtt{P}_{\mathtt{tr}}(y=\omega_c|\boldsymbol{x}) \tag{3.2}$$

The entropy of a discrete probability distribution is an information-theoretic measure of uncertainty[23]. The overall level of *data uncertainty* of the distribution $\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x}, y)$ will be given by the expected conditional entropy:

$$\mathbb{E}_{\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x})}\big[\mathcal{H}[\mathtt{P}_{\mathtt{tr}}(y|\boldsymbol{x})]\big] \tag{3.3}$$

A related way of thinking about *data uncertainty* is to consider the mutual information between y and $\boldsymbol{x}$, defined as:

$$\begin{aligned} \mathcal{I}[y, \boldsymbol{x}] &= \mathtt{KL}[\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x}, y)||\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x})\mathtt{P}_{\mathtt{tr}}(y)] \\ &= \mathcal{H}[\mathtt{P}_{\mathtt{tr}}(y)] - \mathbb{E}_{\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x})}\big[\mathcal{H}[\mathtt{P}_{\mathtt{tr}}(y|\boldsymbol{x})]\big] \end{aligned} \tag{3.4}$$

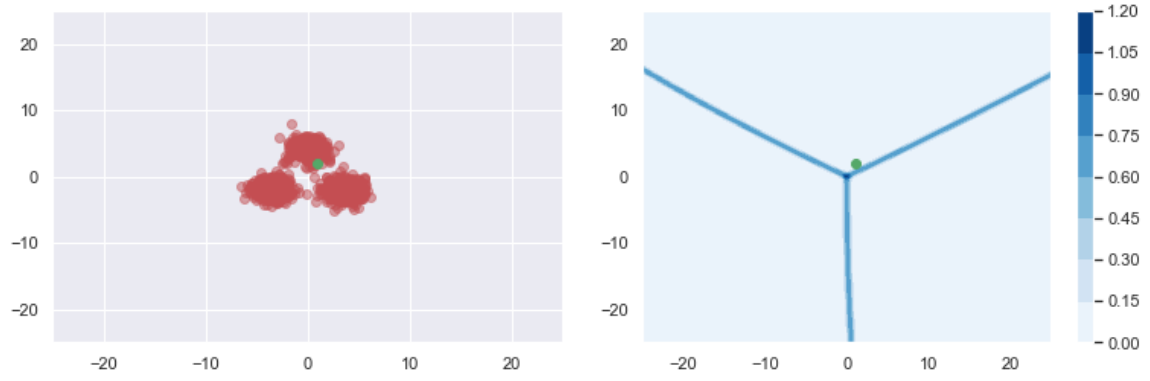
where the marginal distribution $\mathtt{P}_{\mathtt{tr}}(y)$ is given by:

$$\mathtt{P}_{\mathtt{tr}}(y) = \int_{\mathcal{R}^D} \mathtt{p}_{\mathtt{tr}}(\boldsymbol{x}, y) d\boldsymbol{x} \tag{3.5}$$

The mutual information can be interpreted as a measure of *information gain* - it answers the question "how much information does $\boldsymbol{x}$ convey about y?". Alternatively, it can be seen as a measure of independence of y and $\boldsymbol{x}$. If the mutual information is high, then the level of *data uncertainty* is low and $\boldsymbol{x}$ conveys a large degree of information about y and vice versa. Conversely, if the mutual information is 0, then $\boldsymbol{x}$ conveys no information about y, which is another way of saying that $\boldsymbol{x}$ and y are independent. This situation corresponds to a high degree of data uncertainty.

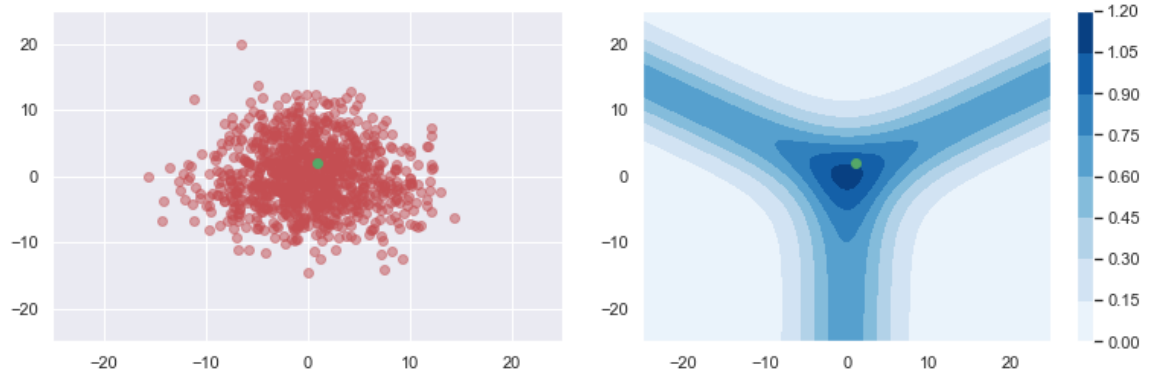
To illustrate the concept of *data uncertainty* more concretely, consider a 'toy' distribution $\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x}, y)$ which consists of three normally distributed clusters with tied isotropic covariances with equidistant means, where each cluster corresponds to a separate class. The marginal distribution over $\boldsymbol{x}$ is given as a mixture of Gaussian distributions:

$$\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x}) = \sum_{c=1}^{3} \mathtt{p}_{\mathtt{tr}}(\boldsymbol{x}|y=\omega_c) \cdot \mathtt{P}_{\mathtt{tr}}(y=\omega_c) = \frac{1}{3}\sum_{c=1}^{3} \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \sigma^2 \cdot \boldsymbol{I}) \tag{3.6}$$



(a) Low data uncertainty (LDU) dataset

(b) Entropy of LDU dataset



(c) High data uncertainty (HDU) dataset

(d) Entropy of HDU dataset

Figure 3.1 The top row depicts the Low Data Uncertainty (LDU) dataset with distinct classes ($\sigma = 1$), where $\mathbb{E}_{\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x})}\big[\mathcal{H}[\mathtt{P}_{\mathtt{tr}}(y|\boldsymbol{x})]\big] = 0.002$ and $\mathcal{I}[y, \boldsymbol{x}] = 1.097$. The bottom row depicts the High Data Uncertainty (HDU) dataset with overlapping classes ($\sigma = 4$), where $\mathbb{E}_{\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x})}\big[\mathcal{H}[\mathtt{P}_{\mathtt{tr}}(y|\boldsymbol{x})]\big] = 0.706$ and $\mathcal{I}[y, \boldsymbol{x}] = 0.393$.

The conditional distribution over the classes y can be obtained via Bayes' rule:

$$\mathtt{P}_{\mathtt{tr}}(y = \omega_c|\boldsymbol{x}) = \frac{\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x}|y = \omega_c) \cdot \mathtt{P}_{\mathtt{tr}}(y = \omega_c)}{\sum_{k=1}^{3} \mathtt{p}_{\mathtt{tr}}(\boldsymbol{x}|y = \omega_k) \cdot \mathtt{P}_{\mathtt{tr}}(y = \omega_k)} = \frac{\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_c, \sigma^2 \cdot \boldsymbol{I})}{\sum_{k=1}^{3} \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_k, \sigma^2 \cdot \boldsymbol{I})} \tag{3.7}$$

Samples of $\boldsymbol{x}$ from the marginal $\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x})$ for the case when $\sigma = 1$ are shown in figure 3.1a. Here, as the three classes are distinct and non-overlapping, it is easy to assign a test sample $\boldsymbol{x}^*$ (green point in figure 3.1a) to the correct class. The conditional entropy, shown in figure 3.1b, is high only along the decision boundaries between classes. The expected conditional entropy is 0.002 and the mutual information between y and $\boldsymbol{x}$ is 1.097. Now consider the dataset in shown in figure 3.1c, where the covariances of each cluster are increased so that there is a large degree of class overlap. The entropy, shown in figure 3.1d, is now high in a wide region

along the decision boundaries and is highest in the area of class overlap. Due to the large degree of class overlap, it will be more difficult to assign the same test sample $\boldsymbol{x}^*$ (green point) to the correct class. The expected conditional entropy is 0.706, which is approximately 45 times larger than for the dataset with no class overlap. The mutual information is 0.393, which is about a third of the mutual information with no class overlap. Clearly, there is low uncertainty in the prediction when the classes are non-overlapping and high uncertainty in the predictions when they do overlap. Thus, the first dataset, with no overlap, will be referred to as the Low Data Uncertainty (LDU) dataset through the rest of this thesis. The artificial dataset with significant class overlap will be referred to as the High Data Uncertainty (HDU) dataset throughout the rest of this thesis. These two datasets will be used to illustrate examples of approaches discussed in this chapter and in chapter 4.

For classification problems *data uncertainty* arises from the natural complexity of the data and the structure of decision boundaries. Datasets which contain a large number of fine-grained classes have a higher level of *data uncertainty*, as the distinctions between classes erodes. To illustrate, consider a dataset with two sets of labels - one with ten 'coarse' classes and one with a hundred fine-grained classes, with ten classes corresponding to different variants of each class from the first data set. The first set of labels contains the classes 'animals, cars, airplanes' and seven other classes. The second set of labels contains the fine-grained classes "dog, wolf, cat, ..." corresponding to the coarse class "animals" and the fine-grained classes "Audi, Ferrari, BMW, Mercedes, etc..." corresponding to the coarse class "cars" and so on. There will be more confusion between different types of cars and different types of animals than there will be between cars and animals. Thus, the second set of labels will have a higher level of *data uncertainty*. Thus, *data uncertainty* depends on how the input space is partitioned into regions belonging to different classes. If the partitioning is such that the regions are distinct, then there is low *data uncertainty*. Conversely, if the regions are partitioned such that certain regions are similar, then there is high *data uncertainty*.

Now consider the situation described in figure 3.2, where the test sample $\boldsymbol{x}^*$ is far away from the region of training data. Such a sample will be referred to, interchangeably, as either an *out-of-distribution* or *out-of-domain* sample because it is sampled from a distribution $\mathtt{p}_{\mathtt{out}}(\boldsymbol{x}, y)$ different to the one from which the training data was sampled. In mathematical terms, the input can be considered out-of-distribution relative to the training data if its probability density under the marginal distribution $\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x}^*)$ is smaller than a threshold δ. Although in practice $\mathtt{p}(\boldsymbol{x})$ is unavailable.

$$\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x}^*) \leq \delta \tag{3.8}$$

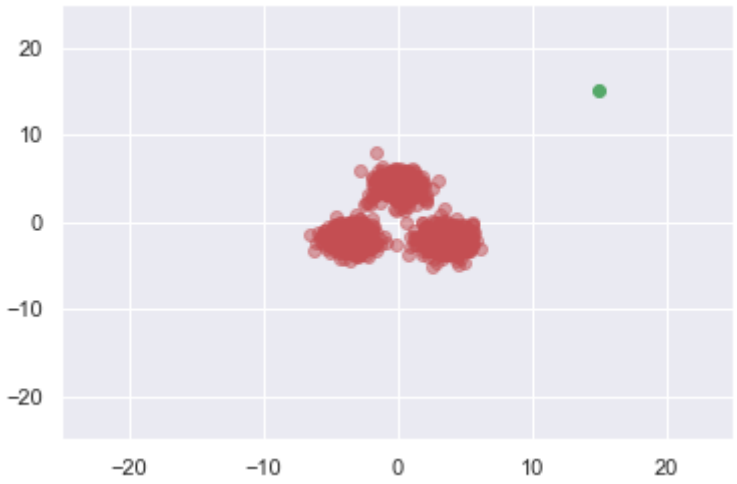


Figure 3.2 Low data uncertainty dataset ($\sigma = 1$) with out-of-distribution input (green dot).

In this situation, a model trained on a finite dataset $\mathcal{D}_{trn}$ may have no understanding or *knowledge*[2] of the discriminative mapping $\boldsymbol{x} \mapsto y$ appropriate to $\mathtt{p}_{\mathtt{out}}(\boldsymbol{x}, y)$, due to distributional mismatch. The test sample $\boldsymbol{x}^*$ could correspond to unseen variations of known classes or to an example of a new, unseen class. As no training data is observed in that region, without strong modelling assumptions it is difficult to know to what class $\boldsymbol{x}^*$ actually belongs. Thus, there will be a high level of *knowledge uncertainty* in the prediction.

3.1.2 Uncertainty for Regression

Having discussed the sources of uncertainty for classification tasks, we now discuss the sources of uncertainty in the context of regression tasks, showing examples of each source of uncertainty on artificial datasets. Consider the following a finite dataset $\mathcal{D}_{trn}$ sample from a distribution $\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x}, \boldsymbol{y})$ over inputs $\boldsymbol{x} \in \mathcal{R}^D$ and targets $\boldsymbol{y} \in \mathcal{R}^K$:

$$\begin{aligned} \mathcal{D}_{trn} &= \{\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\}_{i=1}^{N} \\ \{\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}\} &\sim \mathtt{p}_{\mathtt{tr}}(\boldsymbol{x}, \boldsymbol{y}) \end{aligned} \tag{3.9}$$

Similar to classification tasks, where *data uncertainty* is represented by the entropy of the true posterior over classes (eqn. 3.2), *data uncertainty* for regression tasks is represented by the *Differential Entropy* of the true posterior over targets $\boldsymbol{y}$:

$$\mathcal{H}[\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x})] = -\int_{\mathcal{R}^K} \mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x}) \ln \mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x}) d\boldsymbol{y} \tag{3.10}$$

[2]Subject to generalization ability and prior knowledge encoded by the user.

Differential entropy can be interpreted as a measure of the concentration of probability density on the support. A crucial difference between *data uncertainty* for regression and classification is that *entropy* is non-negative and bounded while *differential entropy* can be unbounded and negative, depending on the nature of $\mathtt{p}(\boldsymbol{y}|\boldsymbol{x})$ and the dynamic range of $\boldsymbol{y}$. [3] In general, differential entropy is not a strict generalization of discrete (Shannon) entropy and lacks many of its properties and interpretations. However, *relative* differential entropy *is* the exact equivalent of discrete KL-divergence for continuous distributions. Thus, while differential entropy cannot be used as an *absolute* measure of uncertainty, it can be used as a *relative* measure of uncertainty. For example, it is impossible to make absolute statements about uncertainty for continuous random variables, but it is possible to say whether there is more or less uncertainty at an input $\boldsymbol{x}_A^*$ than at an input $\boldsymbol{x}_B^*$. In contrast, it is possible to make *both* absolute *and* relative statements about uncertainty for discrete random variables. A more detailed discussion of the properties of entropy and differential entropy is beyond the scope of this thesis and can be found in [23]. As in the case of classification, *data uncertainty* can be assessed via the mutual information between $\boldsymbol{y}$ and $\boldsymbol{x}$, defined as:

$$\begin{aligned}\mathcal{I}[\boldsymbol{y}, \boldsymbol{x}] &= \mathtt{KL}[\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x}, \boldsymbol{y})||\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x})\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y})] \\ &= \mathcal{H}[\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y})] - \mathbb{E}_{\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x})}\big[\mathcal{H}[\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x})]\big]\end{aligned} \tag{3.11}$$

where the marginal distribution $\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y})$ is given by:

$$\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}) = \int_{\mathcal{R}^D} \mathtt{p}_{\mathtt{tr}}(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{x} \tag{3.12}$$

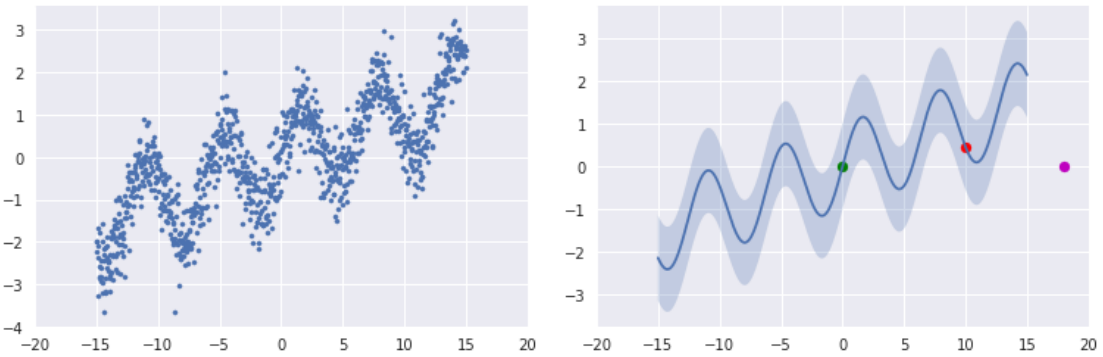
More insight into *data uncertainty* for regression can be obtained by consider an alternative formulation of the generation of the dataset presented in equation 3.9:

$$\begin{aligned}\boldsymbol{x} &\sim \mathtt{p}_{\mathtt{tr}}(\boldsymbol{x}) \\ \boldsymbol{y} &= \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{\epsilon}\end{aligned} \tag{3.13}$$

were $\boldsymbol{\epsilon}$ is additive noise, which itself may be a function of $\boldsymbol{x}$. There are two notable classes of such noise - *homoscedastic noise*, which does not depend on $\boldsymbol{x}$:

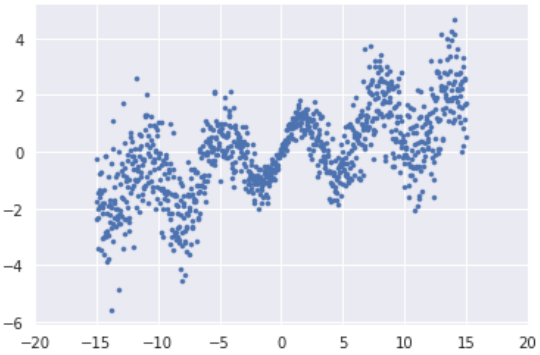
$$\boldsymbol{\epsilon} \sim \mathtt{p}_{\mathtt{tr}}(\boldsymbol{\epsilon}) \tag{3.14}$$

[3] Differential entropy is not scale invariant.

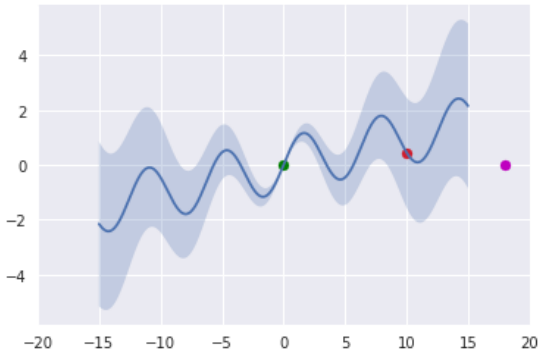


(a) Dataset with homoscedastic noise

(b) Mean and homoscedastic variance



(c) Dataset with heteroscedastic noise



(d) Mean and heteroscedastic variance

Figure 3.3 Figures A and C depict distributions with homoscedastic and heteroscedastic additive Gaussian noise, respectively. Figures B and D depict the decomposition of the dataset into mean and variance. Green and Red points represent inputs in areas of low/high heteroscedastic noise, respectively. Violet point represents out-of-distribution input.

and *heteroscedastic noise*, which is a function of the input $\boldsymbol{x}$:

$$\boldsymbol{\epsilon} \sim \mathrm{p}_{\mathtt{tr}}(\boldsymbol{\epsilon}|\boldsymbol{x}) \tag{3.15}$$

Unlike in classification tasks, where *data uncertainty* arises due to the complexity of the decision boundaries, *data uncertainty* in regression tasks is the result of noisy observations of $\boldsymbol{y}$. This is depicted in figure 3.3. Here either homoscedastic Gaussian noise, shown in figure 3.3a, or heteroscedastic Gaussian noise, shown in figure 3.3c, is added to the function $f(x) = \sin x + \frac{x}{10}$. In figure 3.3b heteroscedastic noise is smallest when $x = 0$ and largest at either end where $|x| = 15$. Note, it is impossible to say whether the noise (and therefore, differential entropy) at any particular point is small or large *in general* - it only possible to compare the noise at two different points in the distribution.

For regression tasks *knowledge uncertainty* represents uncertainty about *both* the mapping represented by the function $\boldsymbol{f}(\boldsymbol{x})$ *and* the nature of the noise $\boldsymbol{\epsilon}$. Unlike classification, where *data uncertainty* and the prediction are linked, *knowledge uncertainty* over the systematic component (prediction) and *knowledge uncertainty* over the noise (*data uncertainty*) can be treated separately for regression tasks. Consider the datasets in figure 3.3. There is no data in the region $|x| \geq 15$. Thus in this region there will be uncertainty in the function which is being modelled as well as the nature of the additive noise. It is difficult to know whether the systematic component changes or continues. Thus, the further a test input $\boldsymbol{x}^*$ is away from the region of training data, the more uncertainty there is in a model's estimation of the *both* the underlying systematic component *and* the noise (*data uncertainty*).

3.2 Estimating Data Uncertainty

Having discussed the nature of *data uncertainty* and *knowledge uncertainty* for both classification and regression in the previous section, we now discuss how to obtain *estimates* of uncertainty in the predictions due to *data uncertainty* for both classification and regression models. Crucially, it is shown how a parametric probabilistic model will naturally estimate *data uncertainty* as a consequence of maximum likelihood estimation, given certain conditions.

3.2.1 Estimating Data Uncertainty for Classification

In order to obtain estimates of *data uncertainty* it is necessary to use a probabilistic model, such as a standard classification neural network which parameterizes a discrete posterior distribution over class labels y conditioned on the input $\boldsymbol{x}$:

$$\begin{aligned} \mathtt{P}(y|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}}) &= \mathtt{Cat}(y; \hat{\boldsymbol{\pi}}) \\ \hat{\boldsymbol{\pi}} &= \boldsymbol{f}(\boldsymbol{x}^*; \hat{\boldsymbol{\theta}}),\ \sum_{c=1}^{K} \hat{\pi}_c = 1,\ \pi_c \geq 0 \end{aligned} \tag{3.16}$$

In chapter 2 section 2.2.1 it was shown that the minimization of the expected negative log-likelihood is equivalent to minimizing the expected KL divergence between the model $\mathtt{P}(y|\boldsymbol{x}; \boldsymbol{\theta})$ and the true conditional distribution $\mathtt{P}_{\mathtt{tr}}(y|\boldsymbol{x})$. This result is reproduced below for

convenience:

$$\begin{aligned}\mathcal{L}^{NLL}(\boldsymbol{\theta}, \mathcal{D}) &= \mathbb{E}_{\mathtt{p_{tr}}(\boldsymbol{x})}\Big[\underbrace{\mathtt{KL}[\mathtt{P_{tr}}(y|\boldsymbol{x})||\mathtt{P}(y|\boldsymbol{x};\boldsymbol{\theta})]}_{Reducible\ Loss} + \underbrace{\mathcal{H}[\mathtt{P_{tr}}(y|\boldsymbol{x})]}_{Irreducible\ Loss}\Big] \\ &\geq \mathbb{E}_{\mathtt{p_{tr}}(\boldsymbol{x})}\Big[\mathcal{H}[\mathtt{P_{tr}}(y|\boldsymbol{x})]\Big]\end{aligned} \tag{3.17}$$

where expected negative log-likelihood is lower-bounded by the expected entropy of $\mathtt{P_{tr}}(y|\boldsymbol{x})$, which, as discussed in section 3.1.1, is the average *data uncertainty* of the distribution $\mathtt{P_{tr}}(y|\boldsymbol{x})$. As was discussed in section 3.1.1, the conditional entropy of the underlying distribution $\mathcal{H}\big[\mathtt{P_{tr}}(y|\boldsymbol{x})\big]$ represents the *data uncertainty* at the input $\boldsymbol{x}$. Thus, a model $\mathtt{P}(y|\boldsymbol{x};\boldsymbol{\theta})$ should capture uncertainty in predictions due to *data uncertainty* in its posterior over classes when trained via maximum likelihood. [4]

A necessary condition for this result to hold true is that an infinite amount of training data is available and that the true underlying distribution lies within the model class which can be parameterized. Thus, in practice a model will only capture *an estimate* of *data uncertainty*, as it is only possible to minimize the KL divergence with respect to an *empirical distribution* derived from a finite training dataset $\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x}, y) = \mathcal{D}_{trn}$. A model $\mathtt{P}(y|\boldsymbol{x};\boldsymbol{\theta})$ may over-fit to the training data and yield poor estimates of uncertainty on a held-out test dataset if it is too large. Alternatively, a model fail to fit the training data at all and also yield poor estimates of *data uncertainty* if it is too simple. The quality of estimates of *data uncertainty* should asymptotically increase with the amount of training data. Models which generalize well should also yield more accurate estimates of *data uncertainty*.

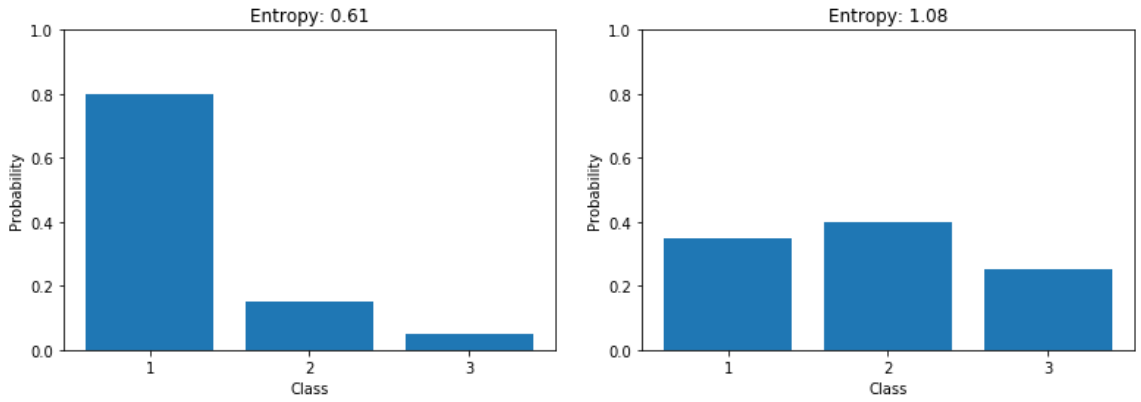
Given a model $\mathtt{P}(y|\boldsymbol{x};\hat{\boldsymbol{\theta}})$ which generalizes well, the expected behavior for inputs which are in regions of low and high data uncertainty are given in figure 3.4: The entropy of the predictive posterior is the model's estimate of *data uncertainty* at a particular test input $\boldsymbol{x}^*$:

$$\mathcal{H}[\mathtt{P}(y|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})] = -\sum_{c=1}^{K} \mathtt{P}(y=\omega_c|\boldsymbol{x}^*;\boldsymbol{\theta}) \ln \mathtt{P}(\omega_c|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}}) \tag{3.18}$$

However, entropy is an 'overall' measure of uncertainty in the predictions as it depends on the entire posterior distribution over classes. It is possible to obtain a measure of uncertainty in predicting *the most likely class* for a particular test input $\boldsymbol{x}^*$ via the *likelihood* of that class:

$$\mathcal{P} = \max_c \{\mathtt{P}(\omega_c|\boldsymbol{x}^*;\boldsymbol{\theta})\} \tag{3.19}$$

[4] In the uncertainty estimation community this seems to be well-known, but unstated.



(a) Low Uncertainty (b) High Uncertainty

Figure 3.4 Indication of uncertainty via posterior over class labels $\mathtt{P}(y|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}})$.

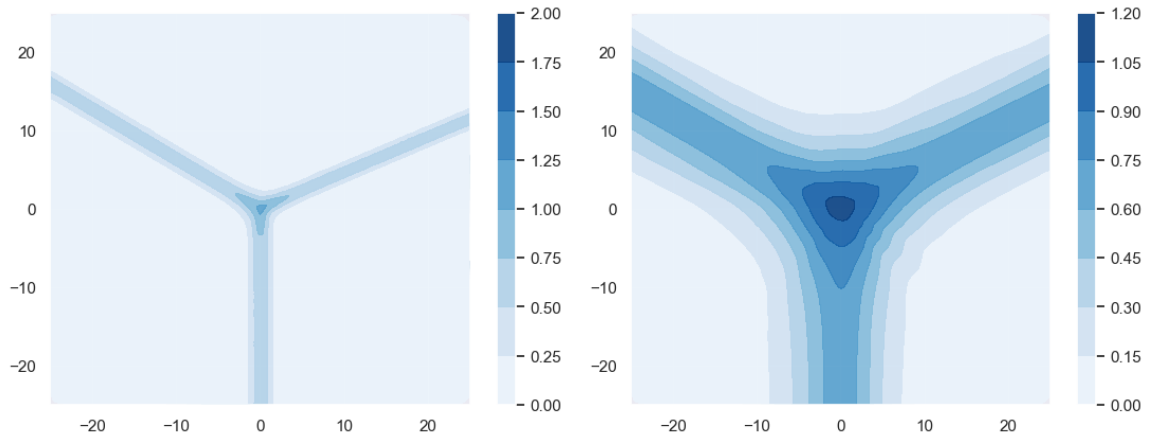
This yields the *confidence of the mode* $\mathcal{P}$[5]. This measure of uncertainty is not affected by the probabilities of the *other classes*, as they are irrelevant to the prediction, and may yield a more precise estimate of uncertainty *in the prediction*.

Estimation of *data uncertainty* is illustrated on the Low Data Uncertainty (LDU) and High Data Uncertainty (HDU) datasets introduced in section 3.1.1. Figure 3.5 demonstrates how the conditional entropy of a pair of simple DNNs trained using maximum likelihood on these captures *data uncertainty*. The distribution of entropy of DNNs trained on these datasets, shown in figures 3.5a and 3.5b, is almost identical to the distribution of entropy of the true underlying distribution, shown in figures 3.1b and 3.1d. In these experiments it is easy to obtain enough training data and the true underlying distribution is easily within the class of models which can be parameterized by the neural networks considered here. In practice it is difficult to fully satisfy these conditions for real applications.

3.2.2 Estimating Data Uncertainty for Regression

The previous section shows how a classification model will naturally capture *data uncertainty* as a consequence of maximum likelihood training. This section will demonstrate how the same can be done for regression tasks when using probabilistic models for regression. It was shown in section 2.2.2 that training a Density Network $\mathtt{p}(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta})$ via maximum likelihood is equivalent to minimizing the expected KL divergence between the model and the true

[5]Here confidence is the inverse of uncertainty.



(a) Low Data Uncertainty dataset (b) High Data Uncertainty dataset

Figure 3.5 Conditional entropy $\mathcal{H}[\mathtt{P}(y|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})]$ of a pair of classification neural networks with 2 hidden layers of 100 units with ReLU activations trained on LDU and HDU datasets with maximum likelihood using Adam [62] optimizer.

conditional distribution $\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x})$. The result is reproduced below for convenience:

$$\begin{aligned}\mathcal{L}^{NLL}(\boldsymbol{\theta},\mathcal{D}) &= \mathbb{E}_{\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x})}\Big[\underbrace{\mathtt{KL}[\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x})||\mathtt{p}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})]}_{Reducible\ Loss} + \underbrace{\mathcal{H}[\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x})]}_{Irreducible\ Loss}\Big] \\ &\geq \mathbb{E}_{\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x})}\Big[\mathcal{H}[\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x})]\Big]\end{aligned} \tag{3.20}$$

where expected negative log-likelihood is lower-bounded by the expected differential entropy of $\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x})$. As discussed in section 3.1.2, *data uncertainty* for regression tasks is expressed as the differential entropy of the underlying distribution. Thus, as the loss is reduced and the model $\mathtt{p}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$ becomes closer to $\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x})$, it will yield increasingly accurate estimates of *data uncertainty* in the form of differential entropy of the density network for a test input $\boldsymbol{x}^*$:

$$\mathcal{H}[\mathtt{p}(\boldsymbol{y}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})] = -\int_{\mathcal{R}^K} \mathtt{p}(\boldsymbol{y}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}}) \ln \mathtt{p}(\boldsymbol{y}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}}) d\boldsymbol{y} \tag{3.21}$$

Modelling *data uncertainty* for regression tasks is more difficult than for classification tasks. For the result in equation 3.20 to hold it is necessary to have infinite training data and for the true distribution to be within the class of models parameterizable by the neural network. To satisfy the second condition it is necessary not only to have a model of sufficient capacity, but to also choose the appropriate probability density function which the Density Network parameterizes. The choice of probability density which the model parameterizes should reflect the structure of *homoscedastic* or *heteroscedastic* noise in the data.

If an incorrect output density is chosen, then the KL divergence term in equation 3.20 cannot be reduced to zero and the model will not fully capture the uncertainty in the data. Unfortunately, it is difficult to a-priori know the structure of the noise in the data and therefore choose the appropriate output density function. However, Mixture Density Networks [10], which can approximate arbitrary distributions $\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x})$ provided they have sufficient mixture components, can be used to overcome this limitation.

Given a well trained density network $\mathtt{p}(\boldsymbol{y}|\boldsymbol{x};\hat{\boldsymbol{\theta}})$ the differential entropy of the posterior over $\boldsymbol{y}$ at a test input $\boldsymbol{x}^*$ will be the model's estimate of uncertainty in the prediction due to *data uncertainty*. However, the expression for differential entropy is not always tractable for continuous probability density functions. Specifically, it is intractable for mixture density networks, though it is possible to obtain approximations [58, 64]. Alternatively, instead of using information-theoretic measures of uncertainty, it is possible to examine second and higher-level moments of $\mathtt{p}(\boldsymbol{y}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})$ in order to characterize the noise via variance, skew, kurtosis, etc... However, depending on the choice of output distribution, information regarding uncertainty may be lost when examining finite moments of a distribution.

3.3 Estimating Knowledge Uncertainty via Single Models

The previous section showed how probabilistic classification and regression models will naturally capture *data uncertainty* as a consequence of maximum likelihood estimation. Unfortunately, it is more difficult to capture estimates of *knowledge uncertainty*, which, as defined in this thesis, is the uncertainty in predictions due to a mismatch between the training and test distributions. In this section we consider approaches to obtaining estimates of *knowledge uncertainty* from a single probabilistic model parameterized using a single set of model parameters[6]. These approaches are discussed in the context of both classification and regression tasks.

3.3.1 Single Model Approaches for Classification

As discussed in section 3.1.1, *knowledge uncertainty* corresponds to uncertainty in predictions due to a lack of knowledge about mapping $\boldsymbol{x} \mapsto y$ in the regions of the input space from which a test sample $\boldsymbol{x}^*$ came from. One way this can be indicated by a model is via a high entropy posterior distribution over class labels (eqn. 3.18), as shown in figure 3.4b. The current section discusses how a single probabilistic classification model can be made to yield

[6] In contrast to ensemble approaches considered in the next section, which use multiple sets of model parameters.

a high entropy posterior for out-of-distribution inputs which come from regions of the input space far from the training data.

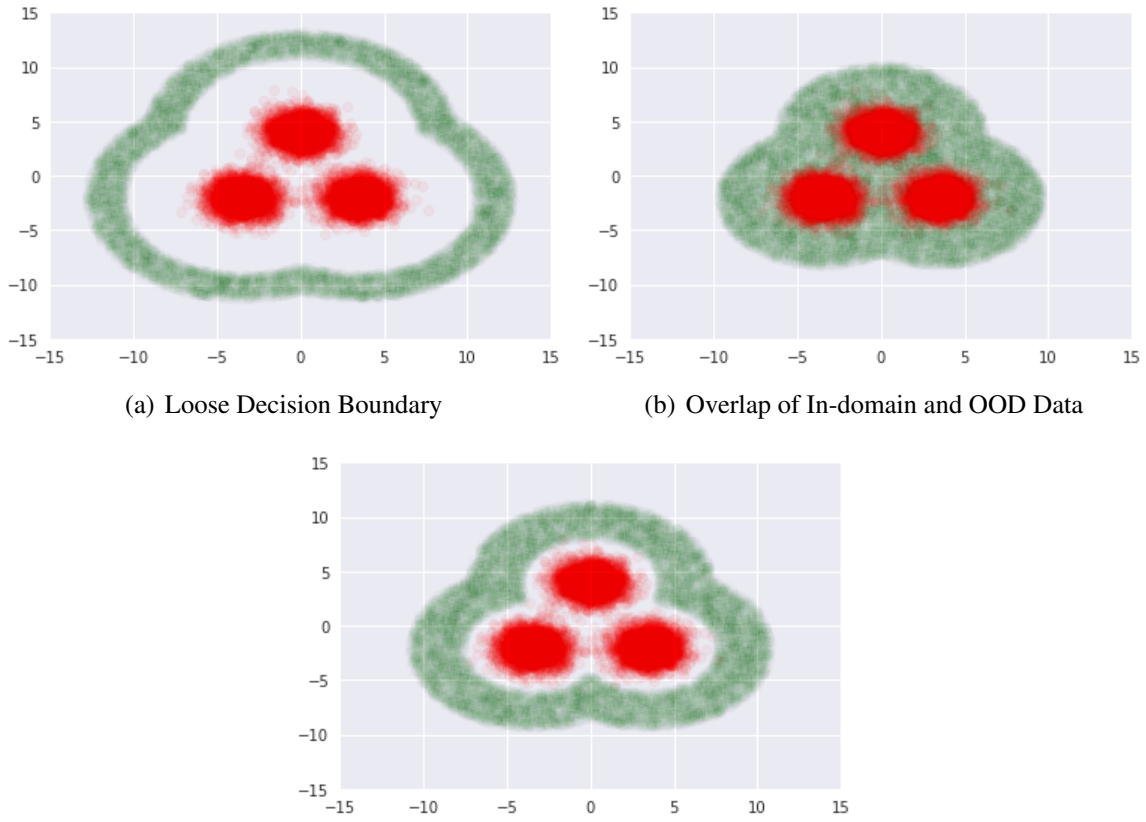
The simplest approach is to simply hope that a model trained via maximum likelihood will naturally yield a high-entropy posterior distribution over classes for out-of-distribution (OOD) inputs. This approach was evaluated as a baseline for detection of misclassifications and out-of-distribution samples in [49]. However, standard maximum likelihood estimation does not contain any mechanism which drives a model to learn the limits of its knowledge. It is, in general, difficult to guarantee any particular behaviour on out-of-distribution data for parametric models trained with standard maximum likelihood, especially neural networks, as they are complex non-linear parametric functions.

A diverse range of approaches has been proposed to modify a neural network classification model to produce high-entropy predictive posteriors for out-of-distribution inputs, such as [73, 74, 91]. However, while many of these methods have impressive empirical results, few have solid theoretical justification for why they work. Consequently, in this thesis only a particular class of single models approaches [73, 80] which does provide a theoretical justification is considered. This approach [73] involves multi-task training of a model to simultaneously minimize negative log-likelihood on in-domain training data and the KL divergence between the model and a uniform distribution $\mathcal{U}(y)$ on out-of-domain training data:

$$\mathcal{L}^{MT}(\boldsymbol{\theta}, \mathcal{D}) = \underbrace{\mathcal{L}^{NLL}(\boldsymbol{\theta}, \mathcal{D}_{trn})}_{In-Domain\ Loss} + \gamma \cdot \underbrace{\mathbb{E}_{\hat{\mathtt{p}}_{out}(\boldsymbol{x})}\left[\mathtt{KL}[\mathcal{U}(y)||\mathtt{P}(y|\boldsymbol{x};\boldsymbol{\theta})]\right]}_{Out-of-Distribution\ Loss} \tag{3.22}$$

where γ is a weight associated with out-of-distribution loss. By minimizing this loss function the model should learn a decision boundary between the in-domain region and the rest of the input space, given an appropriate choice of out-of-distribution training data.

This approach can be interpreted as *explicitly* building in knowledge about the limits of the model's understanding, which is encoded via the choice of out-of-domain training distribution $\mathcal{D}_{out} = \hat{\mathtt{p}}_{out}(\boldsymbol{x})$. It is necessary to select $\mathcal{D}_{out}$ in such a way as to learn a *tight* decision boundary between the in-domain region and everything else. Consider figure 3.6, which depicts the three-class toy classification Low Data Uncertainty dataset introduced in section 3.1.1. The in-domain data is shown in red and out-of-distribution data is shown in green. If the decision boundary is 'too loose', as depicted in figure 3.6a, then certain out-of-domain inputs may be incorrectly considered to be in-domain. Alternatively, if the decision boundary is 'too tight', as depicted in figure 3.6b, then certain in-domain inputs may be incorrectly considered out-of-domain. The out-of-distribution data must be carefully chosen so that it is near the in-domain region, but doesn't overlap with it, as depicted in figure 3.6c. Crucially, the $\mathcal{D}_{out}$ must lie on, or close to, the same manifold on which the in-domain data



(a) Loose Decision Boundary

(b) Overlap of In-domain and OOD Data

(c) Tight Decision Boundary

Figure 3.6 Illustration of in-domain (red) and out-of-domain (green) training data using a toy example. Out-of-domain training data should be close to the in-domain data in order to learn a tight decision boundary around the in-domain region.

lies, as shown in figure 3.7. The motivation for this is that in a real deployment scenario out-of-distribution data is likely going to have similar structure to the in-domain data. For example, in image classification tasks, where an in-domain data consists of natural images, OOD images are also likely to be natural images of the real world. In this scenario, using images of cartoons, random noise or other 'unnatural' images as OOD training data will result in the model learning a decision boundary which is too loose. One approach to generating such data is to use a generative model, such as Factor Analysis [92, 80], Variational Auto-encoders [63] or Generative Adversarial Networks [39, 73]. However, generating appropriate OOD training data is still an open task. In practice it is possible to use data from a different, appropriately chosen dataset [73].

Consider a model trained via the loss specified in equation 3.22 on appropriately chosen out-of-distribution data. The entropy of the posterior of over classes produced by this model becomes a measure of *total uncertainty* rather than just *data uncertainty* or *knowledge*

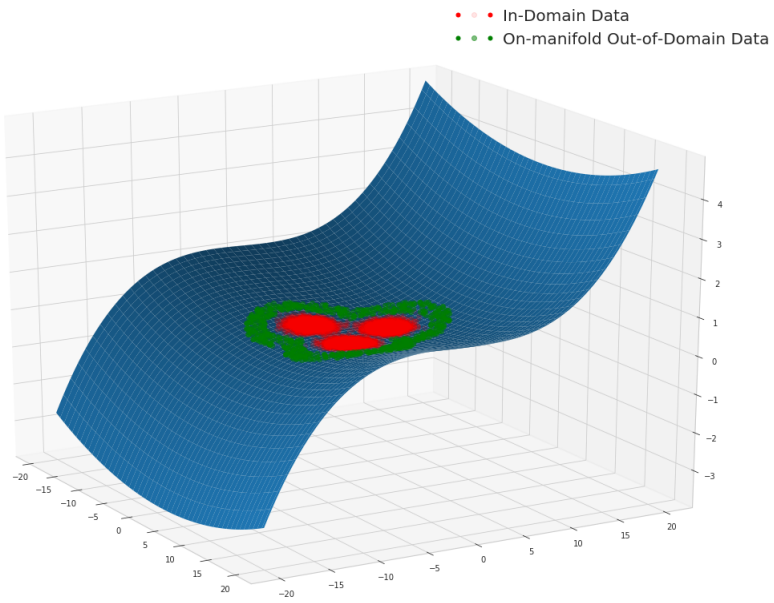
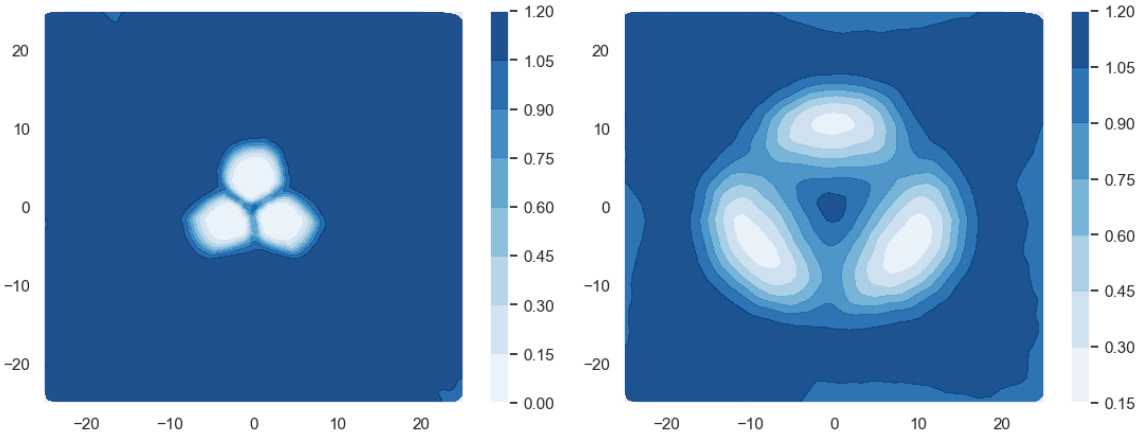


Figure 3.7 Low-dimensional manifold of data in high-dimensional input space. This figure shows both the in-domain data and out-of-distribution data lying on the same 2-dimensional manifold in a 3-D input space.

uncertainty. This leads to a complication - in order to distinguish out-of-domain and in-domain inputs this approach implicitly assumes the entropy of the model's posterior over classes never reaches a maximum in-domain. Otherwise, a high entropy posterior over classes could indicate uncertainty in the prediction due to *either* an in-domain input in a region of severe class overlap or an out-of-distribution input far from the training data. However, this assumption may or may not hold, depending on the nature of the in-domain data. As a result, while it is possible to determine *whether* the model is uncertain, it may not always be possible to robustly determine *why* the model is uncertain using this approach. This is a detriment to applications where it is necessary to determine the source of uncertainty.

This problem is illustrated on the 3-class artificial datasets introduced in section 3.1.1. Figure 3.8 shows the entropy of the predictive posterior derived from a pair of DNNs trained on the Low Data Uncertainty and High Data Uncertainty datasets, respectively, using the loss specified in equation 3.22. The out-of-distribution training data was sampled as shown in figure 3.6c. Figure 3.8 shows that the entropy is high in the entire out-of-domain region on both the LDU and HDU datasets, which is the desired out-of-distribution behaviours. However, the entropy is also high along the decision boundaries, which is most clearly seen in figure 3.8b. In fact, the entropy is equally high in the region where all classes overlap and



(a) Total Uncertainty (b) Total Uncertainty

Figure 3.8 Entropy of predictive posterior $\mathcal{H}[\mathtt{P}(y|\boldsymbol{x};\hat{\boldsymbol{\theta}})]$ derived from DNNs trained in a multi-task fashion on the LDU and HDU datasets using equation 3.22. The DNNs had 2 layers of 100 ReLU units.

out-of-distribution. This makes it difficult to distinguish inputs associated with a high degree of *data uncertainty* from inputs associated with a high degree of *knowledge uncertainty*.

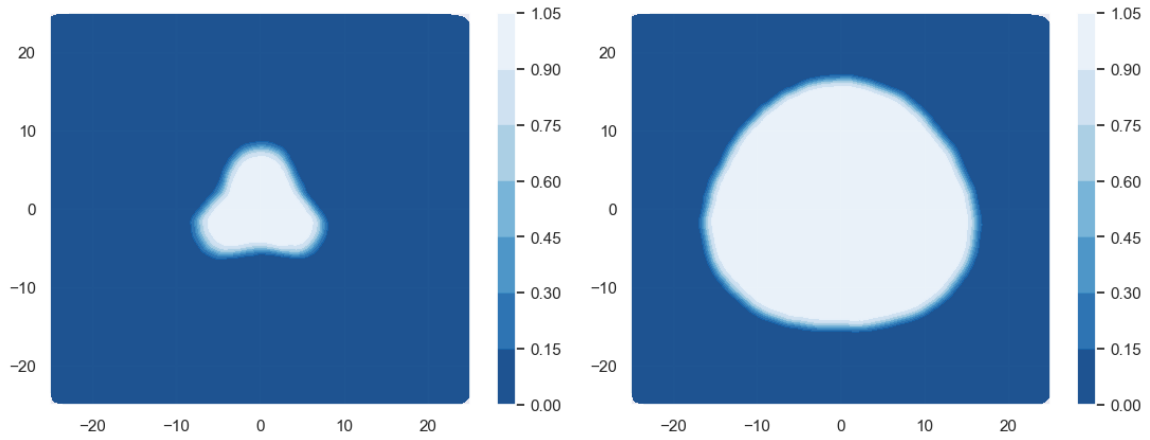
An ad-hoc solution is to introduce an extra 'output head' to yield the probability of the input being in-domain $\mathtt{P}(\mathtt{in}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})$:

$$\begin{aligned} \mathtt{P}(y|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}}) &= \mathtt{Cat}(y;\hat{\boldsymbol{\pi}}) \\ \mathtt{P}(\mathtt{in}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}}) &= \hat{\pi}_{in} \\ \{\hat{\boldsymbol{\pi}},\hat{\pi}_{in}\} &= \boldsymbol{f}(\boldsymbol{x}^*;\hat{\boldsymbol{\theta}}) \end{aligned} \tag{3.23}$$

This model would use the softmax to yield a distribution over classes, from which measures of *total uncertainty* are derived, and a separate probabilistic output head, which gives the probability of the input being in-domain. Such a model can be trained in a multi-task fashion with the following loss:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta},\mathcal{D}) &= \mathcal{L}^{MT}(\boldsymbol{\theta},\mathcal{D}) + \gamma\cdot\mathcal{L}^{AD}(\boldsymbol{\theta},\mathcal{D}) \\ \mathcal{L}^{AD}(\boldsymbol{\theta},\mathcal{D}) &= -\Big(\underbrace{\mathbb{E}_{\hat{\mathtt{p}}_{tr}(\boldsymbol{x})}\big[\ln\mathtt{P}(\mathtt{in}|\boldsymbol{x};\boldsymbol{\theta})\big]}_{In-Domain\ Loss} + \underbrace{\mathbb{E}_{\hat{\mathtt{p}}_{out}(\boldsymbol{x})}\big[\ln\big(1-\mathtt{P}(\mathtt{in}|\boldsymbol{x};\boldsymbol{\theta})\big)\big]}_{Out-of-Distribution\ Loss}\Big) \end{aligned} \tag{3.24}$$

Given this model, if the entropy of the predictive posterior is high and the probability of in-domain is high, then there is a high level of *data uncertainty*. Conversely, if the entropy is high, but probability of in-domain is low, then there is a high level of *knowledge uncertainty*.



(a) Low Data Uncertainty dataset (b) High Data Uncertainty dataset

Figure 3.9 Probability of in-domain input derived from DNNs with an additional output head trained on the LDU and HDU datasets via equation 3.24. The DNNs had 2 layers of 100 ReLU units. Note, the color scale is inverted and white corresponds to high values in this figure.

This approach is illustrated on the 3-class artificial LDU and HDU datasets in figure 3.9, which shows the output of the extra output head for DNNs trained on these datasets using the loss in equation 3.24. The probability of input being in-domain is high in-domain, even in region of class overlap and long decision boundaries, and low elsewhere, which is the desired behaviour. Clearly, this approach alleviates the issues of conflating *data uncertainty* with *knowledge uncertainty*. However, it is not clear how to interpret estimates of *data* and *knowledge* uncertainty within a single consistent probabilistic framework. Specifically, while this approach could work from a practical point of view, it can only answer the question "what is the probability that the input is in-domain?". It does not allow questions about how much uncertainty there is in the prediction of a particular class due to *knowledge uncertainty* to be posed. It is necessary to point out that while figures 3.8 and 3.9 show that it is possible to choose OOD training data which allows the model to learn to yield high estimates of uncertainty out-of-distribution, the task of choosing OOD training data for real tasks and datasets is highly non-trivial.

3.3.2 Single Model Approaches for Regression

Having discussed how to train neural probabilistic models for classification to capture estimates of *knowledge uncertainty*, we now discuss how the same can be done for probabilistic regression models. As discussed in section 3.1.2, *knowledge uncertainty* for regression

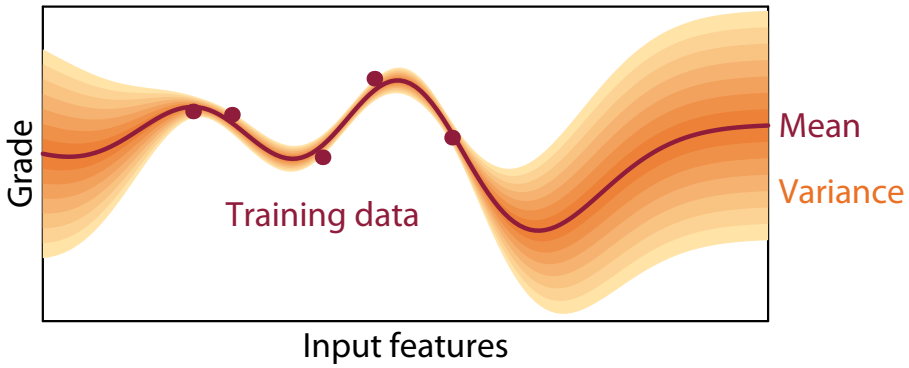


Figure 3.10 Illustration of a toy 1-dimensional Gaussian Process. The variance (uncertainty) increases the further the input is away from the region of training data.

models is uncertainty in both the underlying systematic mapping $\boldsymbol{x} \mapsto \boldsymbol{y}$, represented by the function $\boldsymbol{f}(\boldsymbol{x})$, and the nature of the additive homoscedastic/heteroscedastic noise $\mathrm{p}(\boldsymbol{\epsilon})$ for out-of-domain test inputs $\boldsymbol{x}^*$. Consider figure 3.10 which shows a toy 1-dimensional Gaussian Process, a non-parametric Bayesian model[7], whose variance, an estimate of *knowledge uncertainty*, increases the further the input is from the region on training data. Gaussian Processes, being non-parametric models, do this by measuring an 'average distance' from the test input $\boldsymbol{x}^*$ to the training data. However, neural networks are parametric models and function in a fundamentally different way to Gaussian Processes, so it is not possible to use measures of distance between an input and the training data. The goal is to construct a model $\mathrm{p}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$ which *emulates* the behaviour of a Gaussian Process. Specifically, the model should yield increasing estimates of uncertainty, via differential entropy (eqn. 3.21), for example, as the further the input is away from the region of training data.

Having defined the desired behaviour, let's consider how it can be achieved. As was previously discussed, there are no guarantees on the behavior of a neural network trained via maximum likelihood on out-of-distribution inputs. One way to ensure that a Density Network parameterized by a neural network yields high-entropy posteriors for out-of-distribution inputs is to explicitly train it to do so via multi-task training [80]. Here, a multi-task loss is specified for in-domain training data $\mathcal{D}_{trn}$ and out-of-distribution training data $\mathcal{D}_{OOD}$:

$$\mathcal{L}(\boldsymbol{\theta}, \mathcal{D}) = \mathcal{L}_{in}(\boldsymbol{\theta}, \mathcal{D}_{trn}) + \gamma \cdot \mathcal{L}_{out}(\boldsymbol{\theta}, \mathcal{D}_{OOD}) \tag{3.25}$$

[7]More information about Gaussian Process models can be found in chapter 7.

where γ is the weight of the out-of-distribution loss. Here the in-domain loss is the KL-divergence between the model and a 'teacher model' $\mathtt{p}(\boldsymbol{y}|\boldsymbol{x};\tilde{\boldsymbol{\theta}})$ trained on in-domain data:

$$\mathcal{L}_{in}(\boldsymbol{\theta}, \mathcal{D}_{trn}) = \mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{y},\boldsymbol{x})}\big[\mathtt{KL}[\mathtt{p}(\boldsymbol{y}|\boldsymbol{x};\tilde{\boldsymbol{\theta}})||\mathtt{p}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})]\big] \tag{3.26}$$

The teacher model can be constructed by training a standard Density Network on in-domain data via maximum likelihood. The limitation of this loss function is that if the Density Network is a mixture density network the KL divergence between the teacher and target model can only be exactly computed if they have the same number of mixture components and each component is individually matched. Otherwise, KL-divergence is intractable and an approximation will need to be used. For out-of-distribution data the KL-divergence between the model and high-entropy out-of-domain distribution $\mathtt{p}_{\mathtt{out}}(\boldsymbol{y})$ is minimized:

$$\mathcal{L}_{out}(\boldsymbol{\theta}, \mathcal{D}_{out}) = \mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{out}}(\boldsymbol{x})}\big[\mathtt{KL}[\mathtt{p}_{\mathtt{out}}(\boldsymbol{y})||\mathtt{p}(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})]\big] \tag{3.27}$$

The target out-of-domain distribution $\mathtt{p}_{\mathtt{out}}(\boldsymbol{y})$ should have then same form as the output distribution of the Density Network. One potential choice of $\mathtt{p}_{\mathtt{out}}(\boldsymbol{y})$ is to use the marginal distribution of the training data $\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y})$, as the differential entropy of the marginal $\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y})$ is equal to or larger than the mean differential entropy of the conditional distribution $\mathtt{p}_{\mathtt{tr}}(\boldsymbol{y}|\boldsymbol{x})$, as shown in section 3.1.2.

Conceptually, this approach is similar to the one discussed for classification tasks [73]. The idea is the same - to *explicitly* build in knowledge about the limits of the model's knowledge. In the context of regression this can be interpreted as training a model to emulate a Gaussian Process and yield increasingly high variances further away from the region of training data. Just like the approach discussed in the previous section, this approach requires out-of-distribution training data, sampled from an out-of-domain distribution $\mathtt{p}_{\mathtt{out}}(\boldsymbol{x})$, which may be non-trivial to obtain. This data must be close to the in-domain data and to lie on (or near to) the same data manifold in order to learn to tight decision boundary between the in-domain region and everything else.

However, this approach suffers from a similar issue to the one discussed for classification in the previous section - the differential entropy of the posterior over classes becomes a measure of *total uncertainty*. This means that it is difficult to know a-priori whether the model is yielding a high differential entropy posterior due to having an out-of-distribution input or due to being in a region of particularly high heteroscedastic noise. However, unlike similar approaches for classification, where the entropy of the posterior is bounded, it may be possible to construct a $\mathtt{p}_{\mathtt{out}}(\boldsymbol{y})$ whose differential entropy is far higher for out-of-distribution data

than for in-domain data, depending on the choice of output density function parameterized by the model. While this may work in practice, it is still a heuristic approach. Furthermore, this may adversely affect the model due to such practicalities as having different dynamic ranges for in-domain and out-of-distribution differential entropy. It is possible to apply the same ad-hoc solution of having an extra separate output head to yield probability of in-domain $\mathtt{P}(\mathtt{in}|\boldsymbol{x};\hat{\boldsymbol{\theta}})$. However, just as in the case of classification, it is difficult to interpret the uncertainty estimates within a single unified probabilistic framework.

3.4 Estimating Knowledge Uncertainty via Ensembles

The previous section discussed how knowledge about the limitations of a model's knowledge can be *explicitly* build into a single model $\mathtt{P}(y|\boldsymbol{x};\hat{\boldsymbol{\theta}})$. However, these approaches require out-of-distribution training data and either don't allow sources of uncertainty to be determined, or do so in an ad-hoc fashion. The current section discusses how measures of *knowledge uncertainty* can be obtained by considering an *ensemble* of models $\mathcal{M}$, where each model $\mathcal{M}$ is defined *both* by the model parameters $\boldsymbol{\theta}$ and the architectural choices Λ, such as architecture, initialization, training scheme, etc... Note, that a change of notation is made in this section for generality. This change of notation is for generality of discussion. Here, instead of explicitly building in knowledge into a single mode, ensemble approaches make use of the property that a single model $\mathtt{P}(y|\boldsymbol{x}^*,\mathcal{M}^{(m)})$ displays a particular range of behaviours for in-domain data and has 'undefined' behaviour for out-of-distribution data. An ensemble of *independent* models $\{\mathtt{P}(y|\boldsymbol{x}^*,\mathcal{M}^{(m)})\}_{m=1}^{M}$ is therefore going to be consistent in-domain and yield a diverse set of predictions for inputs which are out-of-distribution, as each model will yield a different 'undefined behaviour'. By using this property of an ensemble, both *data uncertainty* and *knowledge* or *distributional* uncertainty can be assessed within a single consistent probabilistic framework without the need for out-of-distribution training data. The current section discusses ensemble approaches for both classification and regression models, as well as their limitations.

3.4.1 Ensemble Approaches for Classification

In this thesis a *Bayesian* viewpoint on ensembles is adopted, as it provides a particularly elegant probabilistic framework which allows *distributional uncertainty* to be linked to Bayesian *model uncertainty*. The essence of Bayesian methods is to treat the models $\mathcal{M}$ as random variables and place a prior distribution $\mathtt{p}(\mathcal{M})$ over them to compute a posterior

distribution over models $\mathtt{p}(\mathcal{M}|\mathcal{D})$ via Bayes' rule:

$$\begin{aligned}\mathtt{p}(\mathcal{M}|\mathcal{D}) &= \frac{\mathtt{p}(\mathcal{D}|\mathcal{M})\mathtt{p}(\mathcal{M})}{\mathtt{p}(\mathcal{D})} \\ &\propto \mathtt{p}(\mathcal{D}|\mathcal{M})\mathtt{p}(\mathcal{M}) \\ &\propto \mathtt{p}(\mathcal{D}|\boldsymbol{\theta}, \Lambda)\mathtt{p}(\boldsymbol{\theta}|\lambda)\mathtt{P}(\Lambda)\end{aligned} \tag{3.28}$$

Here, *model uncertainty* is captured in the posterior distribution $\mathtt{p}(\mathcal{M}|\mathcal{D})$. The expected predictive distribution for a test input $\boldsymbol{x}^*$ is obtained by taking the expectation with respect to the model posterior:

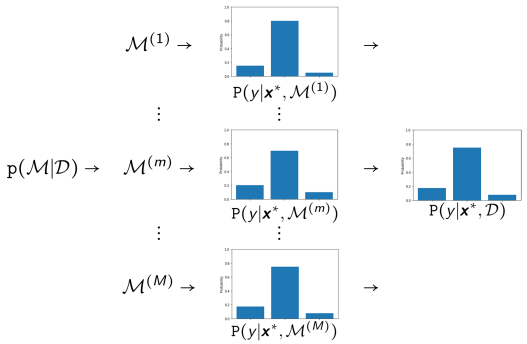
$$\mathtt{P}(y|\boldsymbol{x}^*, \mathcal{D}) = \mathbb{E}_{\mathtt{p}(\mathcal{M}|\mathcal{D})}\big[\mathtt{P}(y|\boldsymbol{x}^*, \mathcal{M})\big] \tag{3.29}$$

In practice it is standard to work with distributions over model parameters $\boldsymbol{\theta}$ rather than architectures. Typically, a particular model architecture $\hat{\lambda}$ is chosen and a posterior distribution over the model parameters $\boldsymbol{\theta}$ is computed:

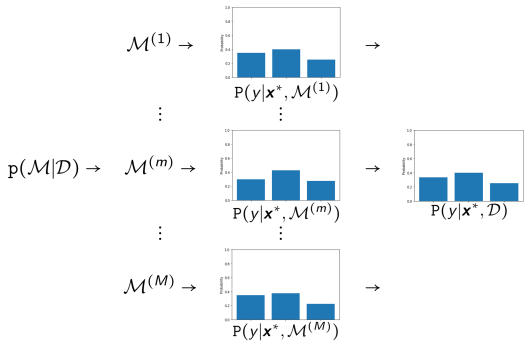
$$\mathtt{p}(\boldsymbol{\theta}|\mathcal{D}, \Lambda = \hat{\lambda}) \propto \ \mathtt{p}(\mathcal{D}|\boldsymbol{\theta}, \Lambda = \hat{\lambda})\mathtt{p}(\boldsymbol{\theta}|\Lambda = \hat{\lambda}) \tag{3.30}$$

However for the sake of generality in this section posteriors over models $\mathcal{M}$ rather than model parameters are considered.

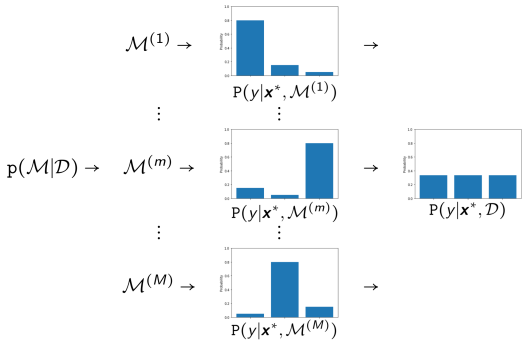
Consider an ensemble $\{\mathtt{P}(y|\boldsymbol{x}^*, \mathcal{M}^{(m)})\}_{m=1}^{M}$ of models sampled from the posterior $\mathtt{p}(\mathcal{M}|\mathcal{D})$. Each of the models $\mathcal{M}$ yields a *different* estimate of *data uncertainty*. Uncertainty in predictions due to *model uncertainty* is expressed as the level of spread, or 'disagreement', of a ensemble sampled from the posterior. The aim is to craft a posterior $\mathtt{p}(\mathcal{M}|\mathcal{D})$, via appropriate choice of prior $\mathtt{p}(\mathcal{M})$, which yields an ensemble that exhibits the set of behaviours described in figure 3.11. Specifically, for an in-domain test input $\boldsymbol{x}^*$ the ensemble should yield a consistent set of predictions with little spread, as described in figure 3.11a and figure 3.11b. In other words, the models should agree in their estimates of *data uncertainty*. On the other hand, for inputs which are different from the training data the models in the ensemble should 'disagree' and produce a diverse set of predictions, as shown in figure 3.11c. Ideally, the models should yield increasingly diverse predictions as input $\boldsymbol{x}^*$ moves further away from the training data. If an input is completely unlike the training data, then the level of disagreement should be significant. Measures of *model uncertainty* will capture *knowledge uncertainty* given an appropriate choice of prior. However, if a prior $\mathtt{p}(\mathcal{M})$ which doesn't yield the behavior in figure 3.11 is chosen, then *model uncertainty* will *not* capture *knowledge uncertainty*. Thus, while single model approaches *explicitly* build



(a) In-domain input with low uncertainty



(b) In-domain input with high data uncertainty



(c) Out-of-distribution input

Figure 3.11 Desired behaviors of an ensemble of classification models. Figures A and B show a consistent ensemble in a region of low/high *data uncertainty*, respectively. Figure C shows a diverse ensemble for an out-of-distribution input.

in knowledge about the limitations of the model's knowledge via training data, ensemble approaches *implicitly* build in this knowledge via choice of prior.

Given an ensemble $\{\mathrm{P}(y|\boldsymbol{x}^*, \mathcal{M}^{(m)})\}_{m=1}^{M}$ which exhibits the desired set of behaviours, the entropy of the expected distribution $\mathrm{P}(y|\boldsymbol{x}^*, \mathcal{D})$ can be used as a measure of *total uncertainty* in the prediction. Effectively $\mathrm{P}(y|\boldsymbol{x}^*, \mathcal{D})$ is a 'single model' whose behaviour out-of-distribution is determined via ensemble diversity rather than out-of-distribution training data. Just as in the case of a single model, it is not possible to determine from the entropy of the predictive posterior whether this uncertainty is due to a high degree of *data* or *model uncertainty*. Howevever, uncertainty in predictions due to *model uncertainty* can be assessed via measures of the spread, or 'disagreement', of the ensemble such as *Mutual Information*:

$$\underbrace{\mathcal{I}[y, \mathcal{M}|\boldsymbol{x}^*, \mathcal{D}]}_{\textit{Model Uncertainty}} = \underbrace{\mathcal{H}\left[\mathbb{E}_{\mathrm{p}(\mathcal{M}|\mathcal{D})}[\mathrm{P}(y|\boldsymbol{x}^*, \mathcal{M})]\right]}_{\textit{Total Uncertainty}} - \underbrace{\mathbb{E}_{\mathrm{p}(\mathcal{M}|\mathcal{D})}\left[\mathcal{H}[\mathrm{P}(y|\boldsymbol{x}^*, \mathcal{M})]\right]}_{\textit{Expected Data Uncertainty}} \tag{3.31}$$

This formulation of mutual information allows the *total uncertainty* to decomposed into *model uncertainty* and *expected data uncertainty*. The entropy of the expected distribution, or *total uncertainty*, will be high whenever the model is uncertain - both in regions of severe class overlap and out-of-domain. However, the difference of the entropy of the expected posterior and the expected entropy of the posterior will be non-zero only if the models disagree. For example, in regions of class overlap *each* member of the ensemble should yield a high entropy posterior (figure 3.11b) - the entropy of the expected and the expected entropy will be similar and mutual information will be low. In this situation *total uncertainty* is dominated by *data uncertainty*. On the other hand, for out-of-domain inputs the ensemble yields diverse posterior distributions over classes such that the expected posterior over classes is near uniform (figure 3.11c) while the expected entropy may be much lower. In this region of input space the models' understanding of data is low and the estimates of *expected data uncertainty* are poor.

An alternative measure of ensemble diversity is the *Expected Pairwise KL-Divergence* between each model in the ensemble. Here the expected KL-divergence between *independent* samples from $\mathtt{p}(\mathcal{M}|\mathcal{D})$ is computed:

$$\begin{aligned}
\underbrace{\mathcal{K}[y, \mathcal{M}|\boldsymbol{x}^*, \mathcal{D}]}_{Model\ Uncertainty} &= \mathbb{E}_{\mathtt{p}(\mathcal{M}_1|\mathcal{D})\mathtt{p}(\mathcal{M}_2|\mathcal{D})}\left[\mathtt{KL}[\mathtt{P}(y|\boldsymbol{x}^*, \mathcal{M}_1)||\mathtt{P}(y|\boldsymbol{x}^*, \mathcal{M}_2)]\right] \\
&= \underbrace{-\sum_{c=1}^{K} \mathbb{E}_{\mathtt{p}(\mathcal{M}_1|\mathcal{D})}[\mathtt{P}(y|\boldsymbol{x}^*, \mathcal{M}_1)]\mathbb{E}_{\mathtt{P}(\mathcal{M}_2|\mathcal{D})}[\ln \mathtt{P}(y|\boldsymbol{x}^*, \mathcal{M}_2)]}_{Total\ Uncertainty} \\
&\quad - \underbrace{\mathbb{E}_{\mathtt{p}(\mathcal{M}|\mathcal{D})}\left[\mathcal{H}[\mathtt{P}(y|\boldsymbol{x}^*, \mathcal{M})]\right]}_{Expected\ Data\ Uncertainty} \\
&\geq \mathcal{I}[y, \mathcal{M}|\boldsymbol{x}^*, \mathcal{D}]
\end{aligned} \tag{3.32}$$

where $\mathtt{p}(\mathcal{M}_1|\mathcal{D}) = \mathtt{p}(\mathcal{M}_2|\mathcal{D})$. Interestingly, this measure is an upper bound on the mutual information and also allows *total uncertainty* to be decomposed into *model uncertainty* and *data uncertainty*. Notably, only estimates of *model* uncertainty differ, while the estimate of *data uncertainty* provided by both decompositions is the same.

So far ensembles have been shown to have desirable *theoretical* properties which give them an advantage over single-model approaches. However, there are several *practical* difficulties when working with ensembles. Firstly, inference using an ensemble is M times more computationally expensive than for single model approaches. Given the expression for the predictive posterior of an ensemble, mutual information and expected pairwise KL-divergence given in equations 3.29, 3.31 and 3.32, one may wish to obtain closed-form

expressions and avoid using an ensemble of models in the first place. However, all these expectations are intractable for neural network models, which is why it is necessary to approximate them via an ensemble in the first place.

The second limitation is that obtaining $\mathtt{p}(\mathcal{M}|\mathcal{D})$ is usually intractable for neural networks. To address this model posterior is approximated using either an explicit or implicit variational approximation $\mathtt{p}(\mathcal{M}|\mathcal{D}) \approx \mathtt{q}(\mathcal{M})$. There is a range of approximate inference approaches. Notably, 'Bayes by Backprop' [13] and 'Probabilistic back-propagation' [51] are two approximate inference approaches which attempt to provide an approximate variational posterior $\mathtt{q}(\boldsymbol{\theta})$. Unfortunately, neither method scales to large tasks. Alternatively some approaches attempt to link stochastic optimization with noise on the gradients to Markov Chain Monte-Carlo sampling from the model posterior $\mathtt{p}(\boldsymbol{\theta}|\mathcal{D})$ via Langevin Dynamics [127]. Another approximate inference scheme is Monte-Carlo dropout [36], which approximates sampling from an implicit variational posterior by doing test-time Monte-Carlo dropout, as described in section 2.2.3. Monte-Carlo Dropout is currently the most widespread, computationally cheap and simplest to implement approximate Bayesian approach to generating an ensemble of models which has been applied to a range of tasks [60, 61].

Unfortunately all of these approaches assume priors of convenience, such as independent Gaussian priors, which do not necessarily yield ensembles with the desired properties. In general, it is difficult to select an appropriate model prior and model architecture to craft a model posterior which induces an ensemble with the desired properties for deep, distributed black-box models with millions of parameters, such as neural networks. Furthermore, the need to approximate the true posterior with an approximation adds further complication, as selection of the desired properties of the ensemble now needs to be incorporated into the approximate inference scheme. This makes it hard to guarantee the desired properties of the ensemble for current state-of-the-art Deep Learning models. Consequently measures of ensemble diversity may fail to accurately capture *knowledge uncertainty* in practice.

It is necessary to point out that while the discussion of ensembles so far has been from a Bayesian viewpoint, it is possible construct ensembles using a range of non-Bayesian approaches. For example, it is possible to *explicitly* construct an ensemble of M models by training on the same data with different random seeds [68] and/or different model architectures. Alternatively, it is possible to generate ensembles via *Bootstrap* approaches [92, 99], where each model is trained on a re-sampled version of the training data. These approaches have the same attributes and the same measures of uncertainty can be derived as for Bayesian ensembles. The difference lies in how the behaviour and diversity of the ensemble is controlled. However, it is similarly difficult to guarantee the diversity of ensembles generated using these non-Bayesian approaches.

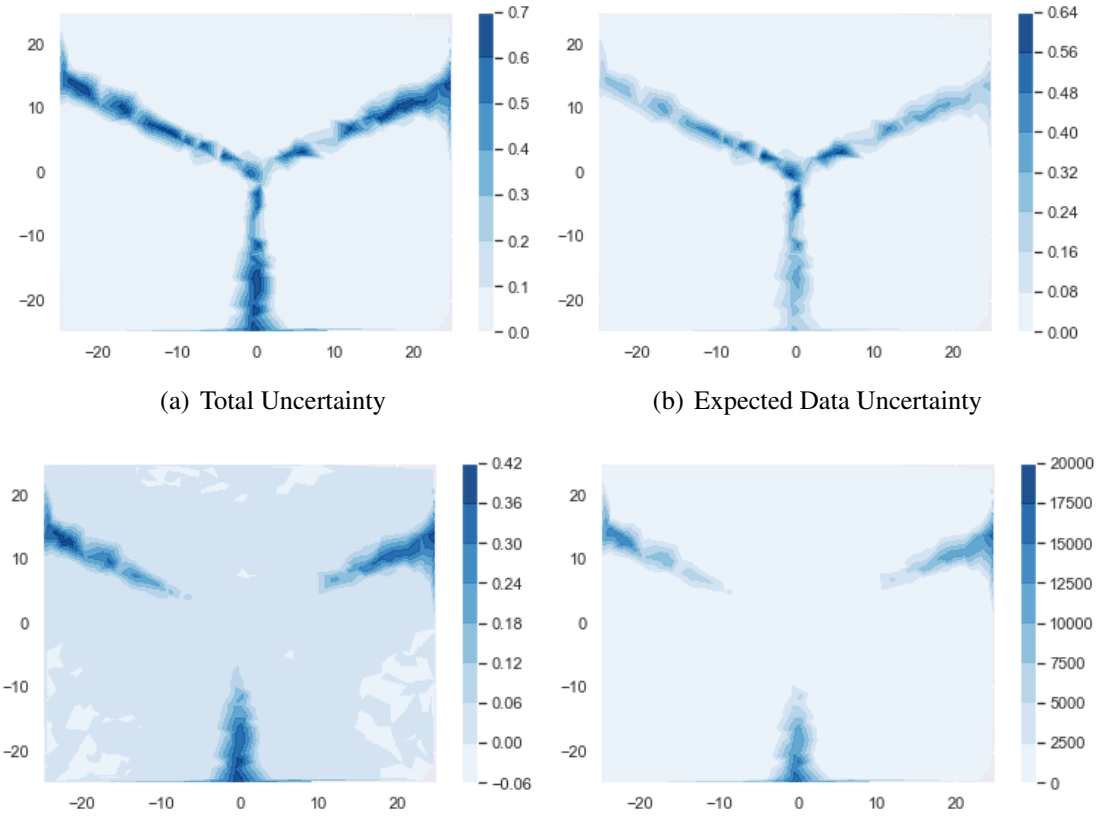
Having discussed the properties and limitations of ensemble approaches in detail, we now illustrate the performance of an ensemble of models trained on the toy 3-class dataset introduced in section 3.1.1. Figure 3.12 shows the behaviour of measures of uncertainty derived from an ensemble of 10 models trained on the low data uncertainty version of the artificial dataset (no class overlap). The total uncertainty, given by the entropy of the expected predictive distribution, is high in a broadening region along the decision boundaries. The expected data uncertainty, given by the average conditional entropy of the models in the ensemble, is highest along the decision boundaries in-domain and decrease further out. The mutual information, obtained via equation 3.31, is high only out-of-distribution and low in-domain. Finally, the expected pairwise KL divergence, derived via equation 3.32, has the same attributes as mutual information, but is significantly higher. Thus, the measures of uncertainty derived from an ensemble naturally decompose into *knowledge uncertainty* and *data uncertainty*.

However, unlike the measures of uncertainty derived from a single model shown in figure 3.8, all measures of uncertainty derived from the ensemble are low out-of-domain far from the decision boundaries. This illustrates that even on toy datasets it is difficult to guarantee the desired behaviour everywhere out-of-distribution [8].

3.4.2 Ensemble Approaches for Regression

Ensemble approaches can be applied to obtain measures of *knowledge uncertainty* for regression tasks in same was as for classification tasks. Consider an ensemble of Density Networks $\{\mathtt{p}(\boldsymbol{y}|\boldsymbol{x}, \mathcal{M}^{(m)})\}_{m=1}^{M}$ sampled from the model posterior $\mathtt{p}(\mathcal{M}|\mathcal{D})$. The goal is to choose a prior $\mathtt{p}(\mathcal{M})$ such that it yields an ensemble which is consistent in-domain and diverse out-of-domain. To illustrate, consider an ensemble of density networks where each model parameterizes a multivariate normal output distribution. This ensemble must be constructed to yield the behaviors described in figure 3.13. Specifically, if the input is in an in-domain region of (relative) low *data uncertainty*, which for regression tasks takes the form of homoscedastic or heteroscedastic noise, then predictions should be consistent and the variance should be small, as shown in figure 3.13a. If the input lies in a region of heavy noise, then the variance should be large, as shown in figure 3.13b. Finally, if then input is out-of-distribution, then the ensemble should be highly diverse as shown in figure 3.13c. Note, that there is diversity *both* in the means $\hat{\boldsymbol{\mu}}^{(m)}$ predicted by the ensemble and the estimates of data uncertainty given by the predicted covariances $\hat{\boldsymbol{\Sigma}}^{(m)}$ of each density

[8] Figures describing behaviour of an ensemble trained on the HDU data and ensembles obtained via Monte-Carlo dropout are omitted here, as they have the same characteristics.



(a) Total Uncertainty

(b) Expected Data Uncertainty

(c) Mutual Information

(d) Expected Pairwise KL-divergence

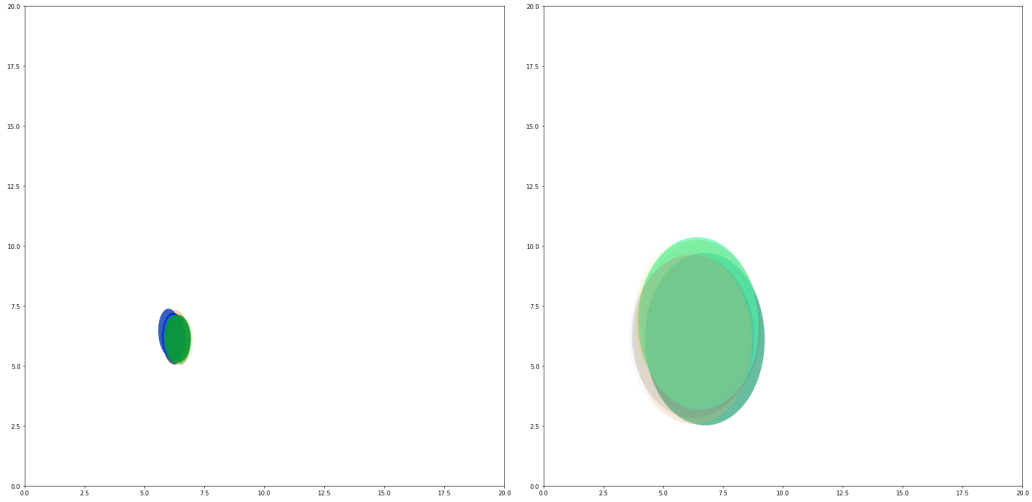
Figure 3.12 Evaluation of measures of uncertainty derived from an ensemble of models trained on the Low Data Uncertainty artificial dataset with maximum likelihood starting from different random initializations. Total Uncertainty, Expected Data Uncertainty and Mutual Information are derived using equation 3.31 and Expected Pairwise KL-divergence using equation 3.32. All models have 2 hidden layers of 100 ReLU units.

network. The effect of *knowledge uncertainty* on both the estimates of *data uncertainty* and the prediction is explicit for regression models, as discussed in section 3.1.2.

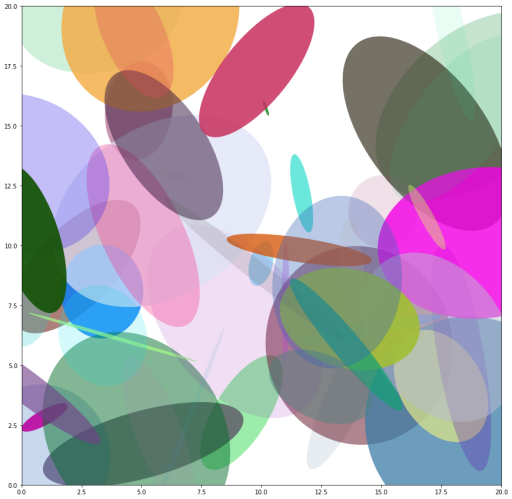
Given an ensembles of density networks, the predictive distribution is obtained by integrating out the model parameters:

$$\mathrm{p}(\boldsymbol{y}|\boldsymbol{x}^*, \mathcal{D}) = \mathbb{E}_{\mathrm{p}(\mathcal{M}|\mathcal{D})}\big[\mathrm{p}(\boldsymbol{y}|\boldsymbol{x}^*, \mathcal{M})\big] \tag{3.33}$$

The differential entropy of the predictive distribution $\mathcal{H}\big[\mathrm{p}(\boldsymbol{y}|\boldsymbol{x}^*, \mathcal{D})\big]$ becomes a measure of *total uncertainty* and has the same attributes as the differential entropy of a single model trained using approaches discussed in the previous section. The diversity of the ensemble



(a) In-domain input with low uncertainty (b) In-domain input with high data uncertainty



(c) Out-of-distribution input

Figure 3.13 Desired an ensemble of regression models which parameterize 2D multivariate normal output distributions. Figures A and B show the means and the variances of the ensemble coincide, while in figure C both the means and the variances are highly diverse.

can be assessed via the mutual information:

$$\underbrace{\mathcal{I}[\boldsymbol{y}, \mathcal{M}|\boldsymbol{x}^*, \mathcal{D}]}_{Model\ Uncertainty} = \underbrace{\mathcal{H}\left[\mathbb{E}_{\mathrm{p}(\mathcal{M}|\mathcal{D})}[\mathrm{p}(\boldsymbol{y}|\boldsymbol{x}^*, \mathcal{M})]\right]}_{Total\ Uncertainty} - \underbrace{\mathbb{E}_{\mathrm{p}(\mathcal{M}|\mathcal{D})}\left[\mathcal{H}[\mathrm{p}(\boldsymbol{y}|\boldsymbol{x}^*, \mathcal{M})]\right]}_{Expected\ Data\ Uncertainty} \tag{3.34}$$

which allows *total uncertainty* to be decomposed into *model uncertainty* and *expected data uncertainty*. Note that although differential entropy is unbounded, the mutual information is bounded from below and is always greater or equal to zero, as it is in fact a KL divergence.

In general, mutual information will be intractable to compute, as the differential entropy of a sum of density functions is intractable. However, if there exists a closed-form solution for the KL-divergence between the output distributions parameterized by the Density Network, then it is possible to consider the expected pairwise KL-divergence between all density networks:

$$\underbrace{\mathcal{K}[\boldsymbol{y}, \mathcal{M}|\boldsymbol{x}^*, \mathcal{D}]}_{Model\ Uncertainty} = \mathbb{E}_{\mathtt{p}(\mathcal{M}_1|\mathcal{D})\mathtt{p}(\mathcal{M}_2|\mathcal{D})}\left[\mathtt{KL}[\mathtt{p}(\boldsymbol{y}|\boldsymbol{x}^*, \mathcal{M}_1)||\mathtt{p}(\boldsymbol{y}|\boldsymbol{x}^*, \mathcal{M}_2)]\right] \tag{3.35}$$

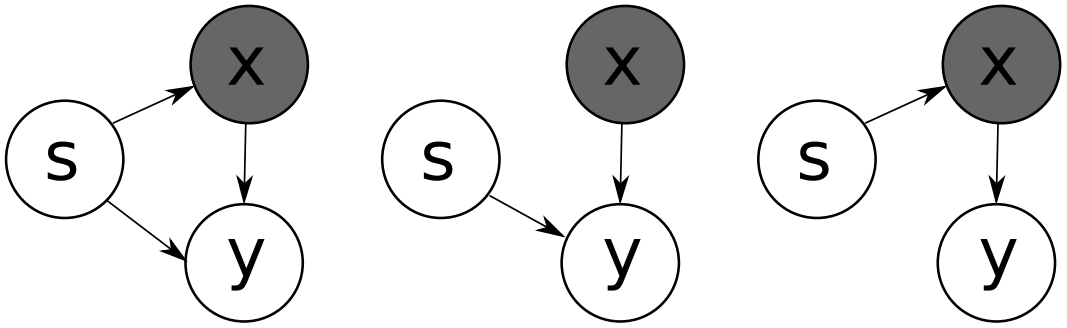
which has the same properties as mutual information. Unfortunately, there is no general closed-form solution for the KL-divergence between Mixture Density Networks, making both the mutual information and the expected pairwise KL-divergence intractable. While it is possible to consider approximations, it is not clear what properties of each measure remain. An alternative is to step away from information theoretic measures of uncertainty and examine second and higher order moments. Consider the law of total variance [28]:

$$\underbrace{\mathbb{V}[\boldsymbol{y}|\boldsymbol{x}^*]}_{Total\ Uncertainty} = \underbrace{\mathbb{V}_{\mathtt{p}(\mathcal{M}|\mathcal{D})}\left[\mathbb{E}_{\mathtt{p}(\boldsymbol{y}|\boldsymbol{x}^*, \mathcal{M})}[\boldsymbol{y}]\right]}_{Model\ Uncertainty} + \underbrace{\mathbb{E}_{\mathtt{p}(\mathcal{M}|\mathcal{D})}\left[\mathbb{V}_{\mathtt{p}(\boldsymbol{y}|\boldsymbol{x}^*, \mathcal{M})}[\boldsymbol{y}]\right]}_{Expected\ Data\ Uncertainty} \tag{3.36}$$

here the total variance of $\mathbb{V}[\boldsymbol{y}|\boldsymbol{x}^*]$ is decomposed into expected variance (*data uncertainty*) and variance of the means of each model in the ensemble (*model uncertainty*). This is conceptually similar to the decomposition available through mutual information, which separates out total uncertainty into model and expected data uncertainties. However, as this only considers second order moments, some information regarding the structure of uncertainty is lost. It is possible to further characterize the uncertainty by consider higher order moments via the *law of total cumulance*, however the resulting expressions may be unwieldly.

3.5 Limits to Modelling Knowledge Uncertainty

As discussed in section 3.1, *knowledge uncertainty*, as defined in this thesis, in the general case will arise due to mismatch between the *joint* distributions $\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x}, y)$ and $\mathtt{p}_{\mathtt{out}}(\boldsymbol{x}, y)$ or between $\mathtt{p}_{\mathtt{tr}}(\boldsymbol{x}, \boldsymbol{y})$ and $\mathtt{p}_{\mathtt{out}}(\boldsymbol{x}, \boldsymbol{y})$ for classification and regression tasks, respectively. This is also known as *dataset shift* and has been studied extensively [102]. Here we examine the nature of this mismatch in greater detail and the limitations to detecting it for discriminative tasks. Consider an unobserved latent variable $\boldsymbol{S}$ which represents the domain and a joint



(a) Domain s affect $\mathtt{p}(\boldsymbol{x}, y)$ (b) Domain s affects $\mathtt{P}(y)$ (c) Domain s affects $\mathtt{p}(\boldsymbol{x})$

Figure 3.14 Relationships between domain variable $\boldsymbol{S}$, inputs $\boldsymbol{x}$ and targets y.

distribution over $\{\boldsymbol{x}, y\}$ which is conditioned on it, such that:

$$\begin{aligned} \mathtt{p}_{\mathtt{tr}}(\boldsymbol{x}, y) &= \mathtt{p}(\boldsymbol{x}, y | \boldsymbol{S} = \boldsymbol{s}_1) \\ \mathtt{p}_{\mathtt{out}}(\boldsymbol{x}, y) &= \mathtt{p}(\boldsymbol{x}, y | \boldsymbol{S} = \boldsymbol{s}_2) \end{aligned} \tag{3.37}$$

This describes the scenario where the domain variable $\boldsymbol{s}$ affects the *joint distribution*. The associated dependency structure between domain $\boldsymbol{S}$ and the variables $\{\boldsymbol{x}, y\}$ is as described in figure 3.14a. In an image classification task, this can correspond to defining a distribution over a different set of images and associated classes altogether. It also possible to consider situations where the domains affects only the marginal distributions over *either* $\boldsymbol{x}$ *or* y, as described in figures 3.14b and 3.14c, respectively.

The situation where the domain only affects $\mathtt{P}(y)$, also known as *prior probability shift* [102], can correspond to changing the distribution over classes, such adding new classes. This is a situation which corresponds to the *task* itself changing. In the context of classification, this is not a situation which the system designer would allow to occur. However, in the context of regression prior probability shift corresponds to the underlying process changing such that the same $\boldsymbol{x}$ now map to a different set of $\boldsymbol{y}$, which is not an impossible situation to occur. Consider predicting the amount of calories burned based on pulse, blood pressure and walking distance - depending on the metabolism of the person, the same distribution of pressure, pulse and distance can correspond to very different levels of calorie expenditure. On the other hand, the situation where the domain affects only $\mathtt{p}(\boldsymbol{x})$, called covariate shift [102], can correspond to the model being exposed to an outlier image which corresponds to new, unseen classes, for example. This can happen often in typical deployment scenarios. There is a range of other, more subtle, forms of dataset shift discussed in [102].However they are outside the scope of the current discussion.

In a real deployment scenario for discriminative models trained on unstructured data (where $\boldsymbol{x} \mapsto y$) neither the domain variable $\boldsymbol{S}$ nor true targets y are observable. This limits the nature of distributional mismatches which are detectable, as it is impossible to detect prior probability shift (shift of the marginal $\mathtt{P}(y)$) based on samples of $\boldsymbol{x}$. Thus, disciminative models are only able detect the distributional mismatches where some form of covariate shift occurs - either when S affects the joint distribution over $\boldsymbol{x}$ and y or only the marginal over $\boldsymbol{x}$.

3.6 Chapter Summary

This chapter discussed the area of estimation of the uncertainty in the predictions of parametric models for classification and regression. Two main sources of uncertainty in predictions were defined - *data uncertainty* and *knowledge uncertainty* in section 3.1. The former relates to the natural noise and class overlap in the data while the latter refers to the distributional mismatch between training and test data. In section 3.2 it was shown that probabilistic discriminative models will naturally capture estimates of *data uncertainty* as a consequence of maximum likelihood training, subject to the conditions that they have sufficient capacity and are exposed to a large amount of training data. In the case of regression models, it is also necessary to either specify the appropriate output density functions for the model to parameterize or to use a mixture density network[10] with a sufficient number of mixture components in order to be able to minimize the reducible loss in equation 3.20.

Two classes of approaches to capturing estimates of *knowledge uncertainty* were discussed in this chapter - single model approaches discussed in section 3.3, and ensemble approaches, discussed in section 3.4. Single model approaches derive uncertainty estimates from a single model's output posterior distribution and/or additional output heads. Crucially, these approaches require extra out-of-distribution training data in order to learn a decision boundary between the in-domain and out-of-domain regions. While it is easy to construct this data for toy tasks, like the Low Data Uncertainty and High Data Uncertainty datasets introduced in section 3.1, construction of out-of-distribution training data for real tasks and datasets is a non-trivial process and an open research question. Ensemble approaches, on the other hand, derive estimates of *knowledge uncertainty* via measures of diversity of an ensemble of models. These approaches allow the sources of uncertainty to be decomposed via measures of uncertainty such as mutual information (equation 3.31) and expected pairwise KL divergence (equation 3.32). The behaviour of an ensemble is controlled, either explicitly or implicitly, via choice of prior over models or model parameters and the approximate inference scheme. Unfortunately, in practice it is difficult to guarantee the desired behavior of an ensemble of neural network models, as was shown in figure 3.12.

Finally, section 3.5 discussed the limitations of modelling *knowledge uncertainty* by discriminative models operating on unstructured data ($\boldsymbol{x} \mapsto y$). Specifically, it is stated that only *knowledge uncertainty* due to a change in the behaviour of the inputs $\boldsymbol{x}$, a situation called *covariate shift*, is detectable, as only the input $\boldsymbol{x}$ is observable in real deployment scenarios.

Chapter 4

Prior Networks

As discussed in the previous chapter, *data uncertainty* is captured naturally as part of maximum likelihood training of probabilistic models, given that certain conditions are satisfied. However, modelling *knowledge uncertainty* is more complicated. The previous chapter discussed single model and ensemble approaches to estimating *knowledge uncertainty* in section 3.3 and 3.4. Single model approaches are easy to train and computationally cheap, but do not allow *data uncertainty* and *knowledge uncertainty* to be assessed separately within a single consistent probabilistic framework. Furthermore, they require out-of-distribution training data in order to define their behaviour for OOD inputs and obtaining such data may be a non-trivial task. On the other hand, ensemble approaches, which derive uncertainty estimates from the diversity of an ensemble of models, have desirable theoretical properties, such as being able to decompose *total uncertainty* into *data uncertainty* and *knowledge uncertainty* and not requiring any out-of-distribution training data. However, ensemble approaches may be significantly computationally more expensive than single model approaches, and it is hard to control the diversity of an ensemble in practice.

The current chapter presents the main theoretical contributions of this thesis - a new class of models called *Prior Networks*. By *directly* parameterizing a conditional *distribution over output distributions*, Prior Networks emulate ensembles, which can be seen as samples from an *implicit* conditional distribution over distributions. This allows Prior Networks to combine the elegant theoretical properties of ensembles with the practical advantages of single model approaches.

This chapter is structured as follows - the general attributes of Prior Networks and measures of uncertainty are discussed in sections 4.1; Prior Networks for classification are discussed in section 4.2; training criteria for classification Prior Networks are investigated in section 4.2.2 and evaluated on the artificial Low Data Uncertainty and High Data Uncertainty datasets from the previous chapter in section 4.2.3; Prior Networks for regression

are discussed in section 4.3; training criteria for regression Prior Networks are discussed in section 4.3.2.

4.1 General Attributes of Prior Networks

Consider an ensemble of classification models $\{\mathtt{P}(y|\boldsymbol{x}^*, \mathcal{M}^{(m)})\}_{m=1}^M$ sampled from the model posterior $\mathtt{p}(\mathcal{M}|\mathcal{D})$. This ensemble can be viewed as an ensemble of categorical output distributions $\{\mathtt{Cat}(y; \boldsymbol{\pi}^{(m)})\}_{m=1}^M$, where the parameters $\boldsymbol{\pi}^{(m)}$ of output distribution are assumed to be deterministic given the architecture, model parameters and input $\boldsymbol{x}^*$:

$$\begin{aligned}\{\mathtt{P}(y|\boldsymbol{x}^*, \mathcal{M}^{(m)})\}_{m=1}^M &= \{\mathtt{Cat}(y|\boldsymbol{\pi}^{(m)})\}_{m=1}^M \\ \boldsymbol{\pi}^{(m)} &= \boldsymbol{f}(\boldsymbol{x}^*; \mathcal{M}^{(m)}), \quad \mathcal{M}^{(m)} \sim \mathtt{p}(\mathcal{M}|\mathcal{D})\end{aligned} \tag{4.1}$$

The crucial insight in this chapter is that this ensemble can equivalently be seen as an ensemble of categorical output distributions $\{\mathtt{Cat}(y; \boldsymbol{\pi}^{(m)})\}_{m=1}^M$ sampled from an *implicit* conditional *distribution over output distributions* $\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}^*, \mathcal{D})$:

$$\{\mathtt{P}(y|\boldsymbol{x}^*, \mathcal{M}^{(m)})\}_{m=1}^M \rightarrow \{\mathtt{Cat}(y; \boldsymbol{\pi}^{(m)})\}_{m=1}^M, \; \boldsymbol{\pi}^{(m)} \sim \; \mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}^*, \mathcal{D}) \tag{4.2}$$

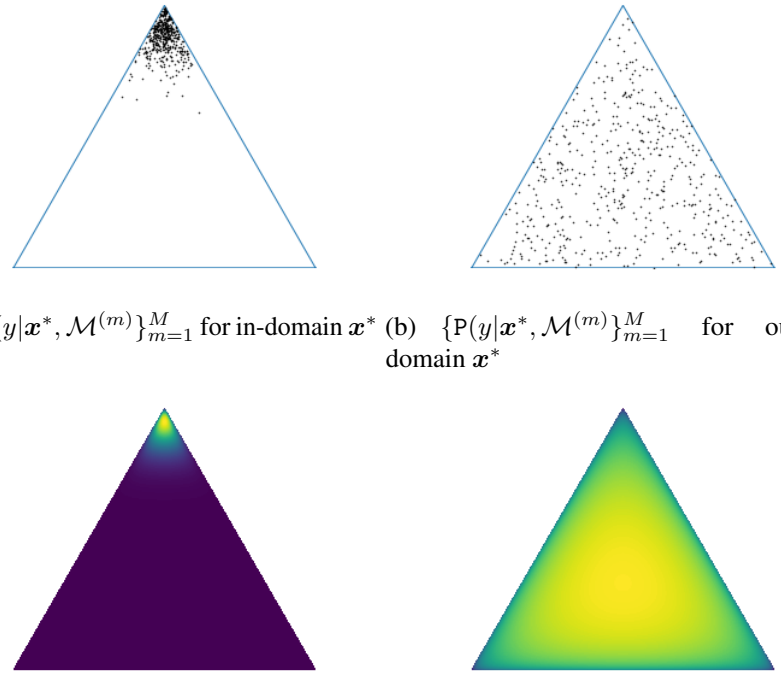
Similarly, ensembles of density networks $\{\mathtt{p}(\boldsymbol{y}|\boldsymbol{x}^*, \mathcal{M}^{(m)})\}_{m=1}^M$ can be interpreted as an ensemble of probability density functions $\{\mathtt{p}(\boldsymbol{y}|\boldsymbol{\mu}^{(m)}, \boldsymbol{\Sigma}^{(m)})\}_{m=1}^M$ sampled from a conditional distribution over output distributions $\mathtt{p}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\boldsymbol{x}^*, \mathcal{D})$. Further intuition can be obtained by considering figure 4.1. The output distributions of an ensemble $\{\mathtt{P}(y|\boldsymbol{x}^*, \mathcal{M}^{(m)}\}_{m=1}^M$ for both in-domain and out-of domain inputs are depicted as points on a $K-1$ dimensional standard *simplex* in figures 4.1a and 4.1b, respectively. These output distributions can be considered to be sampled from a conditional *distribution over output distributions* $\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}^*, \mathcal{D})$, visualized in figures 4.1c and 4.1d.

Using more general notation, ensemble approaches can be seen as sampling output distributions $\mathtt{P}(y|\mathtt{p}_y)$[1] from an *implicit* distribution over output distributions $\mathtt{p}(\mathtt{p}_y|\boldsymbol{x}^*, \mathcal{D})$. This implicit distribution is constructed to yield consistent output distributions $\mathtt{P}(y|\mathtt{p}_y)$ in-domain and diverse output distributions $\mathtt{P}(y|\mathtt{p}_y)$ out-of-domain. This behaviour, in a Bayesian framework, is obtained via appropriate choice of prior distribution $\mathtt{p}(\mathcal{M})$ and approximate inference method. However, controlling the behavior of the implicit distribution over output

[1]This is a general notation for an arbitrary output distribution over discrete variables used in this thesis, and $\mathtt{p}_y$ are parameters of this arbitrary output distribution. An arbitrary output distribution over continuous variables is $\mathtt{p}(\boldsymbol{y}|\mathtt{p}_y)$.

distributions *indirectly* by appropriate selection of prior and approximate inference method is difficult, as discussed in section 3.4.

In this thesis we instead propose to *directly* parameterize a conditional *distribution over output distributions* $\mathtt{p}(\mathtt{p}_\mathtt{y}|\boldsymbol{x}^*, \hat{\boldsymbol{\theta}})$ using a neural network. This class of models will be referred to as *Prior Networks* because the model parameterizes prior distributions over output distributions $\mathtt{P}(y|\mathtt{p}_y)$[2]. Prior Networks will be *explicitly* trained, via methods previously discussed for single model approaches, to emulate the same behaviours of the distribution over output distributions implicit in ensemble methods. This should yield a model which works within the same theoretical framework as ensemble approaches, but has the computational efficiency of single model approaches.



(a) $\{\mathtt{P}(y|\boldsymbol{x}^*, \mathcal{M}^{(m)}\}_{m=1}^M$ for in-domain $\boldsymbol{x}^*$ (b) $\{\mathtt{P}(y|\boldsymbol{x}^*, \mathcal{M}^{(m)}\}_{m=1}^M$ for out-of-domain $\boldsymbol{x}^*$

(c) $\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}^*, \mathcal{D})$ for in-domain $\boldsymbol{x}^*$ (d) $\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}^*, \mathcal{D})$ for out-of-domain $\boldsymbol{x}^*$

Figure 4.1 The predictions of an ensemble for in-domain and out-of-domain inputs are visualized on a simplex and compared to the implicit distribution from they were sampled.

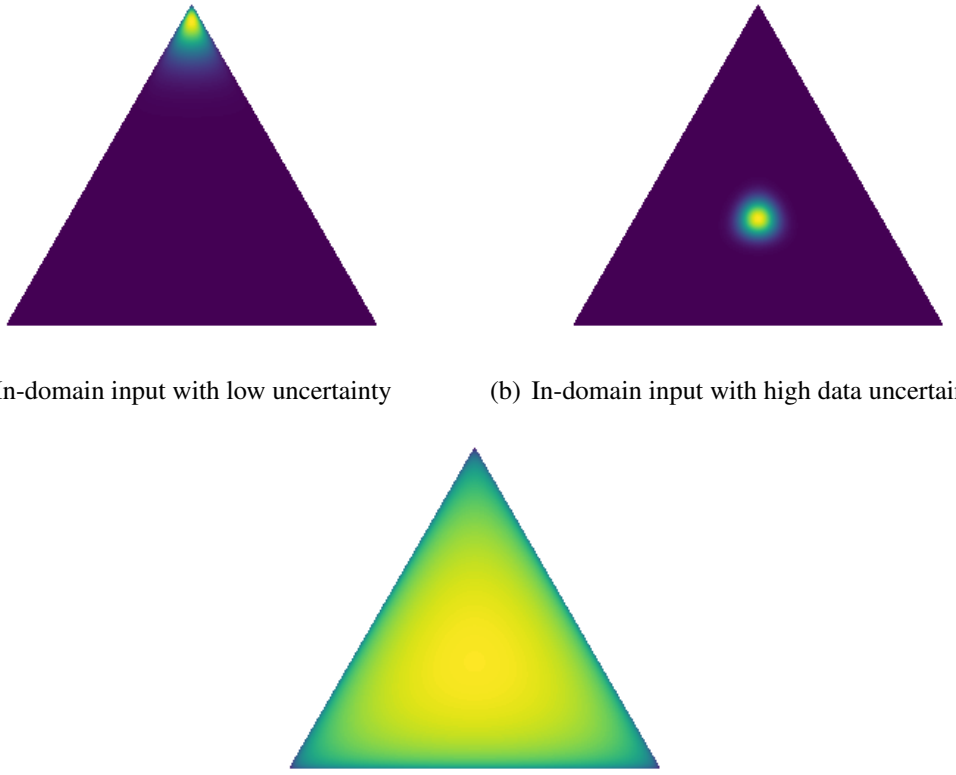
Before discussing Prior Networks specifically for classification and regression, let's consider general properties of Prior Networks in more detail. Prior Networks specify a conditional distribution $\mathtt{p}(\mathtt{p}_\mathtt{y}|\boldsymbol{x}^*, \hat{\boldsymbol{\theta}})$ over point-estimate output distributions $\mathtt{P}(y|\mathtt{p}_y)$. The aim is to emulate the behaviour of an ensemble of models using a single parametric model. Given a Prior Network $\mathtt{p}(\mathtt{p}_\mathtt{y}|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}})$, the predictive distribution will be given by the expected

[2]In hindsight, a more appropriate name would have been Posterior Network.

distribution under the Prior Network:

$$\mathtt{P}(y|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}}) = \mathbb{E}_{\mathtt{p}(\mathtt{p}_\mathtt{y}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})}\left[\mathtt{P}(y|\mathtt{p}_\mathtt{y})\right] \tag{4.3}$$

Estimates of *data uncertainty* are captured by the point-estimate output distributions $\mathtt{P}(y|\mathtt{p}_y)$. Each sample $\mathtt{p}_\mathtt{y}$ represents a different estimate of *data uncertainty* at a particular input $\boldsymbol{x}^*$, similarly to how each model in an ensemble captures a different estimate of *data uncertainty*. *Knowledge uncertainty* is described by measures of spread of $\mathtt{P}(y|\mathtt{p}_y)$, similar to how measures of spread of an ensemble capture *knowledge uncertainty*.



(a) In-domain input with low uncertainty

(b) In-domain input with high data uncertainty

(c) Out-of-distribution input

Figure 4.2 Desired behaviors of a distribution over categorical output distributions.

In order to be able to appropriately capture both *data* and *knowledge* uncertainty an explicit distribution over output distributions parameterized by a Prior Network should have the same properties as the ensembles discussed in section 3.4. Specifically, a Prior Network should yield a sharp distribution $\mathtt{p}(\mathtt{p}_\mathtt{y}|\boldsymbol{x}^*,\hat{\boldsymbol{\theta}})$ over low-entropy output distributions $\mathtt{P}(y|\mathtt{p}_y)$ when it is confident in its prediction. For an input in a region with high degrees of *data uncertainty* a Prior Network should yield a sharp distribution focused on a high-entropy

output distributions $\mathtt{P}(y|\mathtt{p}_y)$, which corresponds to being confident in not being able to properly classify/predict a value due to *data uncertainty*. Finally, for 'out-of-distribution' inputs the Prior Network should yield a high entropy distribution over distributions. For classification models these behaviours can be visualized on a simplex, as shown in figure 4.2. Note, however, that a more nuanced description of in-domain behaviour would be a sharp prior distribution with the mean located on a point on the simplex which corresponds to the appropriate level of *data uncertainty*.

Given a Prior Network which yields the behaviours described above, the *total uncertainty* in the prediction will be given by the entropy (or differential entropy) of the predictive distribution (equation 4.3), just like for single model and ensemble approaches. However, as is the case of ensembles, it is possible to decompose the *total uncertainty* into *data uncertainty* and *knowledge uncertainty* via measures of spread of $\mathtt{P}(y|\mathtt{p}_y)$, such as mutual information:

$$\underbrace{\mathcal{I}[y, \mathtt{p}_\mathtt{y}|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}}]}_{Knowledge\ Uncertainty} = \underbrace{\mathcal{H}\big[\mathbb{E}_{\mathtt{p}(\mathtt{p}_\mathtt{y}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})}[\mathtt{P}(y|\mathtt{p}_\mathtt{y})]\big]}_{Total\ Uncertainty} - \underbrace{\mathbb{E}_{\mathtt{p}(\mathtt{p}_\mathtt{y}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})}\big[\mathcal{H}[\mathtt{P}(y|\mathtt{p}_\mathtt{y})]\big]}_{Expected\ Data\ Uncertainty} \tag{4.4}$$

It is also possible to use the expected pairwise KL divergence between pairs of independent samples from a Prior Network as a measure of spread:

$$\mathcal{K}[\mathtt{p}(\mathtt{p}_\mathtt{y}|\boldsymbol{x}^*, \hat{\boldsymbol{\theta}})] = \mathbb{E}_{\mathtt{p}(\mathtt{p}_\mathtt{y}^{(1)}|\boldsymbol{x}^*,\hat{\boldsymbol{\theta}})\mathtt{p}(\mathtt{p}_\mathtt{y}^{(2)}|\boldsymbol{x}^*,\hat{\boldsymbol{\theta}})}\big[\mathtt{KL}[\mathtt{P}(y|\mathtt{p}_y^{(1)})||\mathtt{P}(y|\mathtt{p}_y^{(2)})]\big] \tag{4.5}$$

where $\mathtt{p}(\mathtt{p}_\mathtt{y}^{(1)}|\boldsymbol{x}^*, \hat{\boldsymbol{\theta}}) = \mathtt{p}(\mathtt{p}_\mathtt{y}^{(2)}|\boldsymbol{x}^*, \hat{\boldsymbol{\theta}})$. As was shown in section 3.3.1, expected pairwise KL divergence is an upper bound on mutual information and can also be decomposed into estimates of *expected data uncertainty* and *knowledge uncertainty*. Finally, it also possible to calculate the differential entropy, a measure of concentration of probability density, of the Prior Network:

$$\mathcal{H}[\mathtt{p}(\mathtt{p}_\mathtt{y}|\boldsymbol{x}^*, \hat{\boldsymbol{\theta}})] = -\int \mathtt{p}(\mathtt{p}_\mathtt{y}|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}}) \ln(\mathtt{p}(\mathtt{p}_\mathtt{y}|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}}))d\mathtt{p}_\mathtt{y} \tag{4.6}$$

As will be shown later, given appropriate choice of prior distribution over categorical distributions or probability density functions, it is possible to calculate all of these measures in closed form without sampling. This makes Prior Networks a non-ad hoc single model approach which is both computationally efficient and operates within a consistent probabilistic framework, combining the best properties of single model and ensemble approaches.

Having discussed the general properties of Prior Networks, it is necessary to discuss how they can be trained. This work considers a method for training Prior Networks based

on approaches for single models described in in chapter 3 section 3.3. A Prior Network is *explicitly* trained in a multi-task fashion to yield the behaviors, described in figure 4.2. Specifically, an in-domain loss $\mathcal{L}_{in}(\boldsymbol{\theta}, \mathcal{D}_{trn})$ is used to train the model to yield low entropy priors $\mathtt{p}(\mathtt{p}_\mathtt{y}|\boldsymbol{x}, \boldsymbol{\theta})$ over either low or high entropy output distributions $\mathtt{P}(y|\mathtt{p}_y)$ for in-domain inputs in regions of low or high *data uncertainty*, respectively. An out-of-distribution loss $\mathcal{L}_{out}(\boldsymbol{\theta}, \mathcal{D}_{out})$ is used to force the model to yield a high-entropy prior distribution $\mathtt{p}(\mathtt{p}_\mathtt{y}|\boldsymbol{x}, \boldsymbol{\theta})$ for out-of-distribution inputs $\boldsymbol{x}^*$. The overall loss will be a weight sum of the in-domain and out-of-distribution losses:

$$\mathcal{L}(\boldsymbol{\theta}, \mathcal{D}) = \mathcal{L}_{in}(\boldsymbol{\theta}, \mathcal{D}_{trn}) + \gamma \cdot \mathcal{L}_{out}(\boldsymbol{\theta}, \mathcal{D}_{out}) \tag{4.7}$$

Like the approaches discussed in section 3.3, this method requires out-of-distribution training data $\mathcal{D}_{out}$. As before, this data can be synthetically generated on the boundary of the in-domain region using generative models such as Factor Analysis, Variational Autoencoders [63] or Generative Adversarial Networks [39]. Alternatively, a different, real dataset can be used as a proxy [73]. While generation of out-of-distribution training data is an open task, the exploration of methods to generate this data is beyond the scope of this thesis. Instead, this work focuses on investigating the in-domain and out-of-distribution losses which yield the desired behaviour of a Prior Network, given appropriate choice of out-of-distribution training data.

4.2 Prior Networks for Classification

Having discussed Prior Networks in general, we now consider how to construct Prior Networks for classification tasks and derive closed-form expressions for the measures of uncertainty discussed in the previous section. A Prior Network for classification parameterizes a conditional prior distribution $\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}})$ over categorical output distributions $\mathtt{Cat}(y|\boldsymbol{\pi})$ using a neural network:

$$\begin{aligned} \mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}}) &= \mathtt{p}(\boldsymbol{\pi}; \hat{\boldsymbol{\alpha}}) \\ \hat{\boldsymbol{\alpha}} &= \boldsymbol{f}(\boldsymbol{x}^*; \hat{\boldsymbol{\theta}}) \end{aligned} \tag{4.8}$$

where $\mathtt{p}(\boldsymbol{\pi}; \hat{\boldsymbol{\alpha}})$ is a prior distribution over categorical distributions.

4.2.1 Parameterization and Uncertainty Measures

There is a range of prior distributions $\mathtt{p}(\boldsymbol{\pi}; \boldsymbol{\alpha})$ which a Prior Network can parameterize, such as the Dirichlet distribution, Generalized Dirichlet Distribution [22, 130], a Mixture of

Dirichlet distributions or the Logistic-Normal distribution [84]. While all of these distribution have support on a standard *simplex*, each yields a different set of behaviours. In this thesis the Dirichlet distribution, a conjugate prior distribution over categorical distributions, is chosen due to its well understood behaviour and tractable analytic properties:

$$\mathtt{p}(\boldsymbol{\pi};\boldsymbol{\alpha}) = \mathtt{Dir}(\boldsymbol{\pi};\boldsymbol{\alpha}) \tag{4.9}$$

The Dirichlet distribution is defined as:

$$\begin{aligned}\mathtt{Dir}(\boldsymbol{\pi};\boldsymbol{\alpha}) &= \mathcal{C}(\boldsymbol{\alpha})\prod_{c=1}^{K}\pi_c^{\alpha_c-1}, \quad \alpha_c > 0\\ \mathcal{C}(\boldsymbol{\alpha}) &= \frac{\Gamma(\alpha_0)}{\prod_{c=1}^{K}\Gamma(\alpha_c)}, \quad \alpha_0 = \sum_{c=1}^{K}\alpha_c\end{aligned} \tag{4.10}$$

where $\Gamma(\cdot)$ is the *gamma function* - a generalization of the factorial function to real numbers. The Dirichlet distribution is parameterized by its concentration parameters $\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_K]^\intercal$. The sum of all the concentration parameters $\alpha_0 = \sum_{c=1}^{K}\alpha_c$ is called the *precision* of the Dirichlet distribution. Higher values of α_0 lead to sharper distributions. The concentration parameters can be interpreted as pseudo-counts of prior occurrences of each class at a given input $\boldsymbol{x}^*$. A Prior Network which parametrizes a Dirichlet distribution will be referred to as a *Dirichlet Prior Network* (DPN).

Given a Dirichlet Prior Network $\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})$ the predictive posterior used for classification is given by the expected output distribution under the Prior Network:

$$\mathtt{P}(y|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}}) = \mathbb{E}_{\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})}\big[\mathtt{P}(y|\boldsymbol{\pi})\big] = \mathtt{Cat}(y|\hat{\boldsymbol{\pi}}) \tag{4.11}$$

where $\hat{\boldsymbol{\pi}}$ is the mean of the Dirichlet distribution. If an exponential output function is used for the DPN, where $\alpha_c = e^{z_c}$, then the expected posterior probability of a label ω_c is given by the softmax function:

$$\mathtt{P}(y=\omega_c|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}}) = \frac{\hat{\alpha}_c}{\sum_{k=1}^{K}\hat{\alpha}_k} = \frac{e^{z_c(\boldsymbol{x}^*)}}{\sum_{k=1}^{K}e^{z_k(\boldsymbol{x}^*)}} \tag{4.12}$$

It turns out, therefore, that standard DNNs for classification with a softmax output function predict the expected categorical distribution under a Dirichlet prior. However, standard DNNs do not necessarily learn to model the distribution around the mean, as the softmax function only predicts the *ratio* of the concentration parameters $\hat{\alpha}_c$ to the precision $\hat{\alpha}_0$ and is not sensitive to arbitrary scaling of the concentration parameters by a multiplicative constant.

The measures of uncertainty of Prior Networks discussed in section 4.1 can be calculated in closed form for Dirichlet Prior Networks. In this section only the final closed-form solutions of the measures of uncertainty are shown. However, the derivations are available in appendix A. The closed-form solution for mutual information is:

$$\begin{aligned}\underbrace{\mathcal{I}[y, \boldsymbol{\pi}|\boldsymbol{x}^*, \hat{\boldsymbol{\theta}}]}_{Knowledge\ Uncertainty} &= \underbrace{\mathcal{H}[\mathbb{E}_{\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{x}^*, \hat{\boldsymbol{\theta}})}[\mathtt{P}(y|\boldsymbol{\pi}]]}_{Total\ Uncertainty} - \underbrace{\mathbb{E}_{\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{x}^*, \hat{\boldsymbol{\theta}})}[\mathcal{H}[\mathtt{P}(y|\boldsymbol{\pi})]]}_{Expected\ Data\ Uncertainty} \\ &= \underbrace{-\sum_{c=1}^{K} \frac{\hat{\alpha}_c}{\hat{\alpha}_0} \ln \frac{\hat{\alpha}_c}{\hat{\alpha}_0}}_{Total\ Uncertainty} \underbrace{- \sum_{c=1}^{K} -\frac{\hat{\alpha}_c}{\hat{\alpha}_0}\big(\psi(\hat{\alpha}_c + 1) - \psi(\hat{\alpha}_0 + 1)\big)}_{Expected\ Data\ Uncertainty}\end{aligned} \tag{4.13}$$

where $\psi(\cdot)$ is the *digamma Function*, which is the derivative of the natural logarithm of the gamma function. Similarly, the closed for solution can also be obtained for the expected pairwise KL divergence:

$$\begin{aligned}\mathcal{K}[\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}})] &= \underbrace{-\sum_{c=1}^{K} \frac{\hat{\alpha}_c}{\hat{\alpha}_0}\big(\psi(\hat{\alpha}_c) - \psi(\hat{\alpha}_0)\big)}_{Total\ Uncertainty} \underbrace{- \sum_{c=1}^{K} -\frac{\hat{\alpha}_c}{\hat{\alpha}_0}\big(\psi(\hat{\alpha}_c + 1) - \psi(\hat{\alpha}_0 + 1)\big)}_{Expected\ Data\ Uncertainty} \\ &= \frac{K-1}{\hat{\alpha}_0}\end{aligned} \tag{4.14}$$

where K is the number of classes. This is a particularly elegant result, as it shows that the *knowledge uncertainty* within a Dirichlet is proportional to the inverse of the precision α_0. Finally, it is also possible to obtain a closed-form solution for the differential entropy of the Dirichlet distribution:

$$\mathcal{H}[\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}})] = -\ln \mathcal{C}(\hat{\boldsymbol{\alpha}}) - \sum_{c=1}^{K} (\hat{\alpha}_c - 1) \cdot \big(\psi(\hat{\alpha}_c) - \psi(\hat{\alpha}_0)\big) \tag{4.15}$$

Differential entropy does not decompose into *data uncertainty* and *knowledge uncertainty*, but is a measure of concentration of the Dirichlet distribution on its support. As discussed in section 3.1.2, differential entropy can in general be unbounded from below and above. However, for the Dirichlet distribution differential entropy is maximum when the distribution is flat over the simplex, which occurs when all the concentration parameters $\boldsymbol{\alpha}$ are equal to 1.

It is interesting to consider an alternative parameterization of Dirichlet Prior Networks. Given equation 4.12, the concentration parameters $\boldsymbol{\alpha}$ of the Dirichlet distribution can be

expressed as the product of the means of the Dirichlet $\hat{\boldsymbol{\pi}}$ and the precision $\hat{\alpha}_0$:

$$\boldsymbol{\alpha} = \hat{\boldsymbol{\pi}} \cdot \alpha_0 \tag{4.16}$$

This leads to an alternative parameterization of the Dirichlet distribution:

$$\begin{aligned} \mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}}) &= \mathtt{p}(\boldsymbol{\pi}; \hat{\boldsymbol{\pi}} \cdot \hat{\alpha}_0) \\ \{\hat{\boldsymbol{\pi}}, \hat{\alpha}_0\} &= \boldsymbol{f}(\boldsymbol{x}^*; \hat{\boldsymbol{\theta}}),\ \hat{\alpha}_0 > 0, \hat{\pi}_c \geq 0, \sum_{c=1}^{K} \hat{\pi}_c = 1. \end{aligned} \tag{4.17}$$

As the precision α_0 is *inversely proportional* to *knowledge uncertainty* (equation 4.14), this form of model is similar to the one described in equation 3.23, which has a extra output head that yields the probability $\mathtt{P}(\mathtt{in}|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}})$. In addition to class probabilities, both the model in equation 3.23 and a Prior Network parameterized as above yield a score of how 'in-domain' the input $\boldsymbol{x}^*$ is. Thus, the model in equation 3.23 can be interpreted as an ad hoc version of a Prior Network. The difference is that the Prior Network approach gives an interpretation to this score which can be directly related to uncertainty in predictions, tying everything together into one consistent probabilistic framework.

4.2.2 Training Criteria

Having discussed how to construct Prior Networks for classification and derived closed-form solutions for the measures of uncertainty described in section 4.1, it is now necessary to discuss how they are trained. The desired set of behaviors of a Prior Network for classification are depicted in figure 4.2. Specifically, a Prior Networks should yield a low-entropy prior focused on low-entropy output distributions for an in-domain input in a region of low *data uncertainty*. For in-domain inputs in a region of high *data uncertainty*, the Prior Network should yield a low-entropy prior focused on high-entropy output distributions. Again, the more nuanced description of in-domain behaviour would be a sharp prior distribution with the mean located on *a point on the simplex which corresponds to the appropriate level of data uncertainty*. Finally, a Prior Network should yield a high-entropy prior over output distributions for out-of-distribution inputs. As described in section 4.1, Prior Networks will be trained in a multi-task fashion using an in-domain loss $\mathcal{L}_{in}(\boldsymbol{\theta}, \mathcal{D}_{trn})$ and an out-of-domain loss $\mathcal{L}_{out}(\boldsymbol{\theta}, \mathcal{D}_{out})$.

Firstly, let's consider how specify an in-domain loss $\mathcal{L}_{in}(\boldsymbol{\theta})$[3] to yield the desired behaviours in figures figures 4.2a and 4.2b. The primary difficulty with specifying an appro-

[3]The dataset $\mathcal{D}_{trn}$ is omitted from the loss function for brevity of notation throughout the rest of this section

priate loss function is that the training data consists of samples from the *true conditional distribution over classes*, but we are interested in learning the appropriate behaviour for a *conditional distribution over distributions over classes*. This means that the training data does not contain information about the desired concentration of the Dirichlet distribution, which should be high in-domain and low out-of-domain. Consider using standard negative log-likelihood loss:

$$\begin{aligned}
\mathcal{L}_{in}^{NLL}(\boldsymbol{\theta}) &= \mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x},y)}\Big[-\sum_{c=1}^{K} \mathcal{I}(y=\omega_c) \ln\big(\mathbb{E}_{\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})}\big[\mathtt{P}(y=\omega_c|\boldsymbol{\pi})\big]\big)\Big] \\
&= \mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x})}\Big[-\sum_{c=1}^{K} \hat{\mathtt{P}}_{\mathtt{tr}}(y=\omega_c|\boldsymbol{x}) \ln \mathtt{P}(y=\omega_c|\boldsymbol{x};\boldsymbol{\theta})\Big] \\
&= \mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x})}\Big[-\sum_{c=1}^{K} \hat{\mathtt{P}}_{\mathtt{tr}}(y=\omega_c|\boldsymbol{x}) (\ln \hat{\pi}_c)\Big] \\
&= \mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x})}\Big[-\sum_{c=1}^{K} \hat{\mathtt{P}}_{\mathtt{tr}}(y=\omega_c|\boldsymbol{x}) (\ln \hat{\alpha}_c - \ln \hat{\alpha}_0)\Big]
\end{aligned} \tag{4.18}$$

As the loss is insensitive to arbitrary scaling of the concentration parameters $\hat{\boldsymbol{\alpha}}$ by a multiplicative constant, the precision α_0 is degenerate if exponential output functions ($\alpha_c = e^{z_c}$) are used. This means that while this loss may appropriately capture *data uncertainty*, it is not necessarily going to yield large values of $\hat{\alpha}_0$ in-domain, as the absolute values of the logits (and therefore the alphas) are not constrained, only their relative magnitudes. This makes this an inappropriate loss for Prior Networks.

A parallel work which investigated a model similar to Dirichlet Prior networks [111] proposed to use an upper bound to the negative log-likelihood obtained via Jensen's inequality:

$$\begin{aligned}
\mathcal{L}_{in}^{NLL}(\boldsymbol{\theta}) &= \mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x})}\Big[-\sum_{c=1}^{K} \hat{\mathtt{P}}_{\mathtt{tr}}(y=\omega_c|\boldsymbol{x}) \ln\big(\mathbb{E}_{\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})}\big[\mathtt{P}(y=\omega_c|\boldsymbol{\pi})\big]\big)\Big] \\
&\leq \mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x})}\Big[-\sum_{c=1}^{K} \hat{\mathtt{P}}_{\mathtt{tr}}(y=\omega_c|\boldsymbol{x}) \mathbb{E}_{\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x};\boldsymbol{\theta})}\big[\ln \mathtt{P}(y=\omega_c|\boldsymbol{\pi})\big]\Big] \\
&= \mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x})}\Big[-\sum_{c=1}^{K} \hat{\mathtt{P}}_{\mathtt{tr}}(y=\omega_c|\boldsymbol{x}) \big(\psi(\hat{\alpha}_c) - \psi(\hat{\alpha}_0)\big)\Big] \\
&= \mathcal{L}_{in}^{NLL-UB}(\boldsymbol{\theta})
\end{aligned} \tag{4.19}$$

This upper bound yields a loss whose form is almost identical to standard negative log-likelihood loss, except with digamma functions instead of natural logarithms. The advantage

of this form is that multiplicative constants will no longer cancel out, which means that the precision is no longer degenerate under any choice of output function. Furthermore, as the magnitude of the concentration parameters grows the difference between the negative log-likelihood and the upper bound decreases. This property can be analyzed by considering the following asymptotic series approximation to the digamma function:

$$\psi(x) = \ln x - \frac{1}{2x} + \mathcal{O}(x^2) \tag{4.20}$$

$$\approx \ln x - \frac{1}{2x} \tag{4.21}$$

Given this approximation, it is easy to show that this upper-bound loss is equal to the negative log-likelihood plus an extra term which drives the concentration parameter $\hat{\alpha}_c$ to be as large as possible:

$$\begin{aligned}
\mathcal{L}_{in}^{NLL-UB}(\boldsymbol{\theta}) &= \mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x})}\Big[-\sum_{c=1}^{K} \hat{\mathtt{P}}_{\mathtt{tr}}(y = \omega_c|\boldsymbol{x})\big(\psi(\hat{\alpha}_c) - \psi(\hat{\alpha}_0)\big)\Big] \\
&\approx \mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x})}\Big[-\sum_{c=1}^{K} \hat{\mathtt{P}}_{\mathtt{tr}}(y = \omega_c|\boldsymbol{x})\big(\ln(\hat{\pi}_c) - \frac{1-\hat{\pi}}{2\hat{\alpha}_c}\big)\Big] \\
&= \mathcal{L}_{in}^{NLL}(\boldsymbol{\theta}) + \mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x})}\Big[\sum_{c=1}^{K} \hat{\mathtt{P}}_{\mathtt{tr}}(y = \omega_c|\boldsymbol{x})\big(\frac{1-\hat{\pi}}{2\hat{\alpha}_c}\big)\Big]
\end{aligned} \tag{4.22}$$

Clearly, this extra term goes to zero and the difference between the upper-bound and the negative log-likelihood vanishes as the concentration increases. This suggests that this loss should correctly capture *data uncertainty* when $\hat{\boldsymbol{\alpha}}$ have a large magnitude, and *knowledge uncertainty* should be captured appropriately due to this loss driving the precision $\hat{\alpha}_0$ to be high. However, a practical issue reported in [111] was that the precision $\hat{\alpha}_0$ has a tendency to grow large and cause numerical overflow, requiring additional ad hoc terms which prevent that from happening.

As these two losses do not yield the desired behaviour of the Dirichlet distribution in a controlled fashion, an obvious alternative to consider is to explicitly define the desired Dirichlet distribution $\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{\beta})$ for every training input $\boldsymbol{x}$, where $\boldsymbol{\beta}$ are the target concentration parameters, and then minimize the KL-divergence between it and the model. This loss can be directly optimized without sampling as the expression of the KL-divergence between two Dirichlet distributions parameterized by concentration parameters $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ is available in

closed-form:

$$\texttt{KL}[\texttt{p}(\boldsymbol{\pi}|\boldsymbol{\beta})||\texttt{p}(\boldsymbol{\pi}|\boldsymbol{\alpha})] = \ln \mathcal{C}(\boldsymbol{\beta}) - \ln \mathcal{C}(\boldsymbol{\alpha}) + \sum_{c=1}^{K}(\beta_c - \alpha_c)\big(\psi(\beta_c) - \psi(\beta_0)\big) \tag{4.23}$$

The target concentration parameters should reflect the desired in-domain behaviour of a Prior Network: a sharp Dirichlet distribution with the mean located on a point which reflects the level of data uncertainty. However, as the only information available to construct this target distribution is the class label, we can only set the target concentration parameters $\boldsymbol{\beta}^{(c)}$ such that the Dirichlet is sharp in the corner corresponding to the target class, and low elsewhere:

$$\beta_k^{(c)} = \begin{cases} \beta + 1 & if\ c = k \\ 1 & if\ \ c \neq k \end{cases} \tag{4.24}$$

Here β should take on a large value, for example $1e2$. Note, the concentration parameters have to be strictly positive, so it is not possible to set them to 0. Instead, they are set to one, which additionally provides a small degree smoothing, preventing the class probabilities from taking on extremely small values. As the Dirichlet is a conjugate prior, the target concentration parameters can be interpreted to represent a *posterior* Dirichlet distribution after β observations of the target class, where the *prior* Dirichlet distribution was flat. This leads to the following loss function, which is what we used in the original work on Prior Networks [81]:

$$\mathcal{L}_{in}^{KL}(y, \boldsymbol{x}, \boldsymbol{\theta}) = \sum_{c=1}^{K} \mathcal{I}(y = \omega_c) \cdot \texttt{KL}[\texttt{p}(\boldsymbol{\pi}|\boldsymbol{\beta}^{(c)})||\texttt{p}(\boldsymbol{\pi}|\boldsymbol{x}; \boldsymbol{\theta})] \tag{4.25}$$

Unfortunately, this loss will not appropriately estimate *data uncertainty*. Consider taking the expectation of this loss with respect to the empirical distribution $\hat{\texttt{p}}_{\texttt{tr}}(\boldsymbol{x}, y) = \mathcal{D}_{trn}$:

$$\begin{aligned}
\mathcal{L}_{in}^{KL}(\boldsymbol{\theta}) &= \mathbb{E}_{\hat{\texttt{p}}_{\texttt{tr}}(\boldsymbol{x},y)}\Big[\sum_{c=1}^{K} \mathcal{I}(y = \omega_c) \cdot \texttt{KL}[\texttt{p}(\boldsymbol{\pi}|\boldsymbol{\beta}^{(c)})||\texttt{p}(\boldsymbol{\pi}|\boldsymbol{x}; \boldsymbol{\theta})]\Big] \\
&= \mathbb{E}_{\hat{\texttt{p}}_{\texttt{tr}}(\boldsymbol{x})}\Big[-\sum_{c=1}^{K} \hat{\texttt{P}}_{\texttt{tr}}(y = \omega_c|\boldsymbol{x}) \int_{\mathcal{S}^{K-1}} \texttt{p}(\boldsymbol{\pi}|\boldsymbol{\beta}^{(c)}) \ln \texttt{p}(\boldsymbol{\pi}|\boldsymbol{x}; \boldsymbol{\theta}) d\boldsymbol{\pi}\Big] \\
&= \mathbb{E}_{\hat{\texttt{p}}_{\texttt{tr}}(\boldsymbol{x})}\Big[-\int_{\mathcal{S}^{K-1}} \sum_{c=1}^{K} \hat{\texttt{P}}_{\texttt{tr}}(y = \omega_c|\boldsymbol{x}) \texttt{p}(\boldsymbol{\pi}|\boldsymbol{\beta}^{(c)}) \ln \texttt{p}(\boldsymbol{\pi}|\boldsymbol{x}; \boldsymbol{\theta}) d\boldsymbol{\pi}\Big] \\
&= \mathbb{E}_{\hat{\texttt{p}}_{\texttt{tr}}(\boldsymbol{x})}\Big[\texttt{KL}\big[\sum_{c=1}^{K} \hat{\texttt{P}}_{\texttt{tr}}(y = \omega_c|\boldsymbol{x}) \texttt{p}(\boldsymbol{\pi}|\boldsymbol{\beta}^{(c)})||\texttt{p}(\boldsymbol{\pi}|\boldsymbol{x}; \boldsymbol{\theta})\big]\Big]
\end{aligned} \tag{4.26}$$

This shows that this loss is effectively minimizing the KL-divergence between the model and a *mixture of Dirichlet distributions* which has a mode in each corner of the simplex. When the level of *data uncertainty* is low, this is not a problem, as there will be only a single mode. However, when there is a significant amount of *data uncertainty* the target distribution will be multi-modal. As the KL-divergence is *zero-avoiding* [11, 92], it will drive the model to spread itself over each mode. The Dirichlet distribution can yield an (almost) multi-modal behaviour with the probability density being high only at the corners of the simplex by driving the precision to be less than the number of classes, as shown in figure 4.3a. However, this is an undesirable behaviour with regards to modelling *data uncertainty*, as the model should instead yield a distribution with a single mode at the center of the simplex as shown in 4.3b.

(a) Actual behaviour (b) Desired behaviour

Figure 4.3 Actual and desired behaviors of Dirichlet distribution in areas of high *data uncertainty* when trained with loss specified in equation 4.25.

The main issue of the KL-divergence loss is that the target distribution $\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{\beta}^{(c)})$ is arithmetically summed in expectation, which induces an incorrect target distribution. Thus, the question is, what is the appropriate loss function such that the correct target Dirichlet distribution, which is sharp at the point on the simplex which corresponds to the correct level of *data uncertainty*, is induced *in expectation*? It turns out that this can be achieved by considering the *reverse* KL-divergence between the target distribution $\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{\beta}^{(c)})$ and the model instead of the *forward* KL-divergence.

$$\mathcal{L}_{in}^{RKL}(y, \boldsymbol{x}, \boldsymbol{\theta}) = \sum_{c=1}^{K} \mathcal{I}(y = \omega_c) \cdot \mathtt{KL}[\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{x}; \boldsymbol{\theta})||\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{\beta}^{(c)})] \tag{4.27}$$

By taking the expectation of the reverse KL-divergence with respect to the empirical distribution, it can be shown that this loss becomes the reverse KL-divergence between the model

and a *geometric mixture of target Dirichlet distributions*.

$$\begin{aligned}
\mathcal{L}_{in}^{RKL}(\boldsymbol{\theta}) &= \mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x})}\Big[\sum_{c=1}^{K}\hat{\mathtt{P}}_{\mathtt{tr}}(y=\omega_c|\boldsymbol{x})\mathtt{KL}\big[\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x};\boldsymbol{\theta})||\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{\beta}^{(c)})\big]\Big] \\
&= \mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x})}\Big[\mathbb{E}_{\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x};\boldsymbol{\theta})}\big[\ln\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x};\boldsymbol{\theta}) - \sum_{c=1}^{K}\hat{\mathtt{P}}_{\mathtt{tr}}(y=\omega_c|\boldsymbol{x})\ln\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{\beta}^{(c)})\big]\Big] \\
&= \mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x})}\Big[\mathbb{E}_{\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x};\boldsymbol{\theta})}\big[\ln\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x};\boldsymbol{\theta}) - \ln\prod_{c=1}^{K}\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{\beta}^{(c)})^{\hat{\mathtt{P}}_{\mathtt{tr}}(y=\omega_c|\boldsymbol{x})}\big]\Big] \\
&= \mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x})}\Big[\mathtt{KL}\big[\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x};\boldsymbol{\theta})||\mathtt{p}(\boldsymbol{\pi}|\bar{\boldsymbol{\beta}})\big]\Big] + C \\
\bar{\boldsymbol{\beta}} &= \sum_{c=1}^{K}\hat{\mathtt{P}}_{\mathtt{tr}}(y=\omega_c|\boldsymbol{x})\cdot\boldsymbol{\beta}^{(c)}
\end{aligned} \tag{4.28}$$

A geometric mixture of Dirichlet distributions results in a standard Dirichlet distribution whose concentration parameters $\bar{\boldsymbol{\beta}}$ are an arithmetic mixture of the target concentration parameters for each class. When there is low *data uncertainty* this loss simply yields the reverse KL-divergence to a sharp Dirichlet at a particular corner. However, when the *data uncertainty* is significant, this loss minimizes the reverse KL-divergence to a Dirichlet with a single mode close to the center of the simplex. This is exactly the behaviour which the model should learn when there is an in-domain input in a region of significant *data uncertainty*. A full derivation is available in appendix A.

It is interesting to further analyze the properties of this loss by decomposing the reverse KL-divergence into the reverse cross-entropy and the negative differential entropy:

$$\mathcal{L}_{in}^{RKL}(\boldsymbol{\theta}) = \mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x})}\Big[\underbrace{\mathbb{E}_{\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x};\boldsymbol{\theta})}\big[-\ln\mathtt{Dir}(\boldsymbol{\pi}|\bar{\boldsymbol{\beta}})\big]}_{Reverse\ Cross{-}Entropy} - \underbrace{\mathcal{H}\big[\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x};\boldsymbol{\theta})\big]}_{Differential\ Entropy}\Big] \tag{4.29}$$

Let's consider the reverse-cross entropy term in more detail:

$$\begin{aligned}
\mathcal{L}_{in}^{RCE}(\boldsymbol{\theta}) &= \mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x})}\Big[\mathbb{E}_{\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x};\boldsymbol{\theta})}\Big[-\sum_{c=1}^{K}\sum_{k=1}^{K}\hat{\mathtt{P}}_{\mathtt{tr}}(y=\omega_c|\boldsymbol{x})\big(\beta_k^{(c)}-1\big)\ln\pi_k\Big]\Big] \\
&= \mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{tr}}(\boldsymbol{x})}\Big[-\sum_{c=1}^{K}\sum_{k=1}^{K}\hat{\mathtt{P}}_{\mathtt{tr}}(y=\omega_c|\boldsymbol{x})\big(\beta_k^{(c)}-1\big)\big(\psi(\hat{\alpha}_k)-\psi(\hat{\alpha}_0)\big)\Big]
\end{aligned} \tag{4.30}$$

When the target concentration parameters $\boldsymbol{\beta}^{(c)}$are defined as in equation 4.24, the form of the reverse KL-divergence loss will be:

$$\begin{aligned}\mathcal{L}_{in}^{RKL}(\boldsymbol{\theta}) &= \mathbb{E}_{\hat{\mathrm{p}}_{\mathtt{tr}}(\boldsymbol{x})}\Big[-\beta \sum_{c=1}^{K} \hat{\mathrm{P}}_{\mathtt{tr}}(y=\omega_c|\boldsymbol{x})\big(\psi(\hat{\alpha}_c) - \psi(\hat{\alpha}_0)\big) - \mathcal{H}\big[\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{x};\boldsymbol{\theta})\big]\Big] \\ &= \beta \cdot \mathcal{L}_{in}^{NLL-UB}(\boldsymbol{\theta}) - \mathbb{E}_{\hat{\mathrm{p}}_{\mathtt{tr}}(\boldsymbol{x})}\Big[\mathcal{H}\big[\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{x};\boldsymbol{\theta})\big]\Big]\end{aligned} \tag{4.31}$$

This is the upper bound on the negative log-likelihood defined in equation 4.19, weighted by the target concentration β and regularized by maximizing the differential entropy of the model. Thus, minimizing the reverse KL-divergence is equivalent to using the upper-bound loss with the appropriate regularization term to prevent it from driving the precision of the model to extreme values. Curiously, this expression is equivalent to the *forward* cross-entropy between discrete distributions, plus additional regularization terms which drive $\hat{\alpha}$ to have appropriate behaviour. In contrast, the expression for the *forward* KL-divergence between Dirichlet distributions contains within itself an expression for the *reverse* cross-entropy between discrete distributions. This property is explored in appendix B.

An advantage of the *reverse* KL-divergence loss is the ability to accurately control the desired precision of the Prior Network in different in-domain region. Consider a situation example where a certain class in under-represented relative to the other classes in the training dataset. It is possible to correct the balance the dataset by over-sampling data-points belonging to that class, but indicate a higher level of *knowledge uncertainty* via a lower target concentration β than for the other classes. Due to the nature of the reverse KL-divergence loss, the concentration parameters will be appropriately smoothed on the boundaries between each class. This suggests that the reverse KL divergence is the optimal loss which allows a Prior Network to learn to yield a sharp distribution at either a corner or near the center of the simplex for in-domain inputs in areas of low or high *data uncertainty*, respectively.

Finally, an alternative approach to training Dirichlet Prior Networks, which is not explored in this thesis and is left to future work, would be to explicitly distill an ensemble of models into a Prior Network via using approaches like teacher-student training [54]. Ideally, this approach should be able to capture both the *data uncertainty* and *knowledge uncertainty* estimates of an ensemble and may yield improvements to classification performance along with a reduction in computational expense. However, it is not clear if a Dirichlet distribution sufficient to exactly capture the behaviour of an ensemble of models.

Having obtained the loss which yields the desired in-domain behaviour of a Prior Network, it is now necessary to discuss the choice of out-of-domain loss. As previously stated, the desired behaviour of a Prior Network for out-of-distribution inputs is a high entropy prior

distribution over categorical distributions where all categorical distributions are equiprobable, as described in figure 4.2c. The maximum entropy Dirichlet distribution is a flat Dirichlet distribution which can be obtained by setting all the concentration parameters to one ($\boldsymbol{\beta} = \mathbf{1}$). Thus, the obvious loss to consider is the KL divergence between the Prior Network and a target Dirichlet distribution $\mathrm{p}(\boldsymbol{\pi}; \boldsymbol{\beta})$ where $\boldsymbol{\beta} = \mathbf{1}$:

$$\mathcal{L}_{out}^{KL}(\boldsymbol{\theta}) = \mathbb{E}_{\hat{\mathrm{p}}_{\text{out}}(\boldsymbol{x})}\left[\mathtt{KL}[\mathrm{p}(\boldsymbol{\pi}; \mathbf{1})||\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{x}; \boldsymbol{\theta})]\right] \tag{4.32}$$

However, as the in-domain loss with the most desirable properties is the *reverse* KL divergence, it makes sense to be consistent and also use the *reverse* divergence as the out-of-distribution loss:

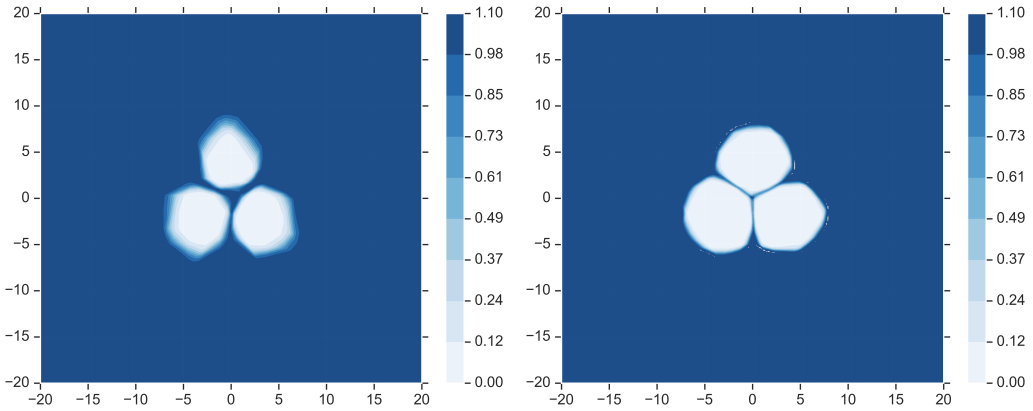
$$\mathcal{L}_{out}^{RKL}(\boldsymbol{\theta}) = \mathbb{E}_{\hat{\mathrm{p}}_{\text{out}}(\boldsymbol{x})}\left[\mathtt{KL}[\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{x}; \boldsymbol{\theta})||\mathrm{p}(\boldsymbol{\pi}; \mathbf{1})]\right] \tag{4.33}$$

which is exactly equivalent to maximizing the differential entropy of the Dirichlet distribution. This also has the advantage that on the boundary of the in-domain and out-of-distribution regions the concentration parameters are appropriately smoothed, and multi-task loss is *always* minimizing the reverse KL-divergence to a standard Dirichlet distribution.

4.2.3 Experiments on Artificial Data

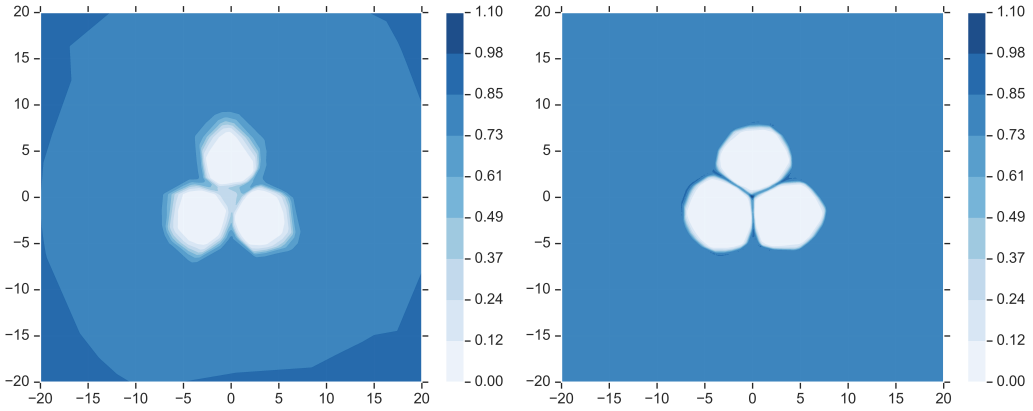
The previous section investigated the theoretical properties of several training criteria for Prior Networks. In this section the properties of these criteria are assessed empirically by using them to train Prior Networks on the artificial 3-class Low Data Uncertainty (LDU) and High Data Uncertainty (HDU) datasets introduced in the chapter 3 section 3.1.1. Specifically, both the forward and reverse KL-divergence between the Prior Network and a target Dirichlet distribution are considered. In these experiments, Prior Networks parameterize the Dirichlet distribution by directly yielding the concentration parameters $\hat{\boldsymbol{\alpha}}$. The models use the same architecture and training hyper-parameters as the previous networks trained on these datasets in chapter 3. The out-of-distribution training data $\mathcal{D}_{out}$ was sampled such that it forms a thin shell around the training data, as shown in figure 3.6c. The target Dirichlet concentration parameters $\boldsymbol{\beta}^{(c)}$ were constructed as described in equation 4.24, with $\beta = 1e3$. The in-domain loss and out-of-distribution losses were equally weighted when trained using the forward KL-divergence loss. However, it was found that it is necessary to weight the out-of-distribution loss 10 times as much as the in-domain loss when using reverse KL divergences.

Figures 4.4 and 4.5 depict the behaviour of measures of uncertainty derived from Prior Networks trained using either the *forward* or the *reverse* KL-divergence loss on the LDU and HDU datasets, respectively. Specifically, the figures depict *total uncertainty*, *expected data*



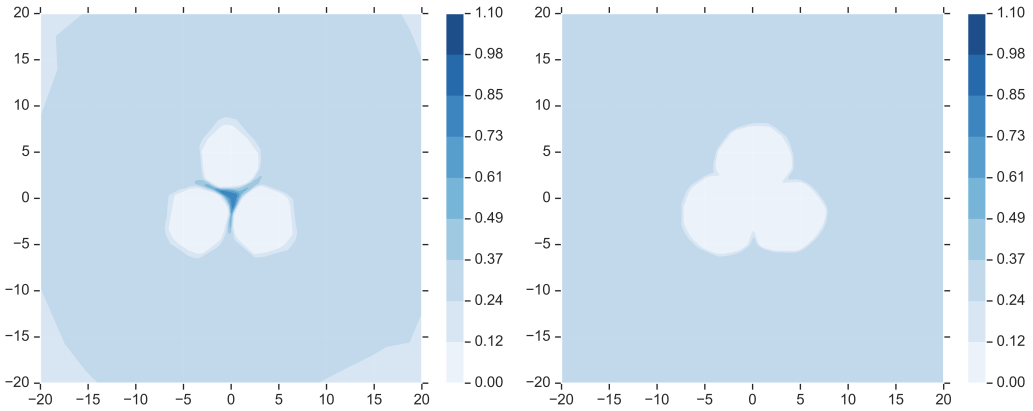
(a) Total Uncertainty - KL

(b) Total Uncertainty - reverse KL



(c) Expected Data Uncertainty - KL

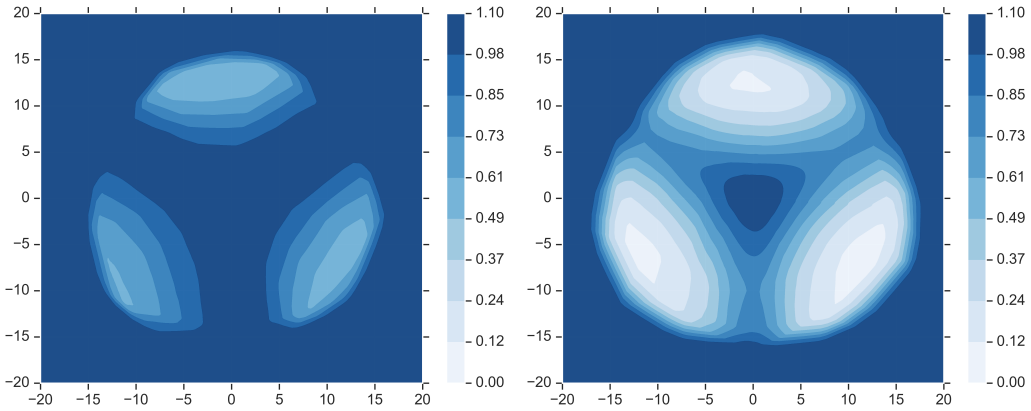
(d) Expected Data Uncertainty - reverse KL



(e) Mutual Information - KL

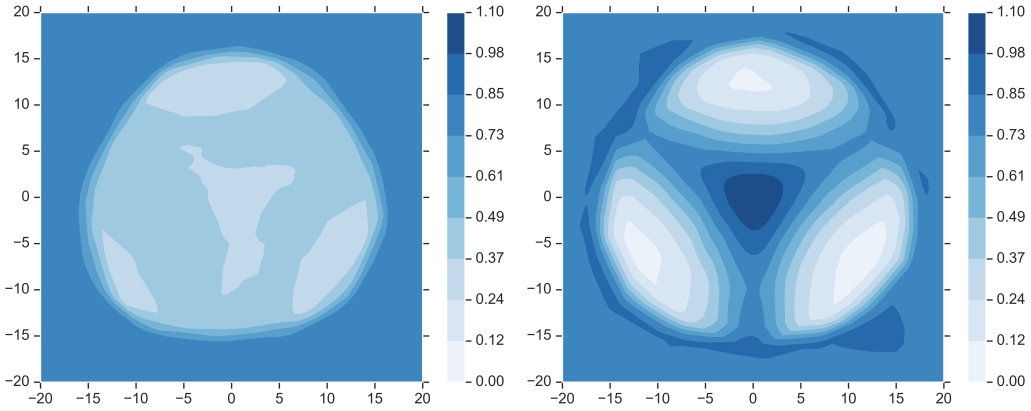
(f) Mutual Information - reverse KL

Figure 4.4 Comparison of measures of uncertainty derived from Prior Networks trained with *forward* and *reverse* KL-divergence loss on the Low Data Uncertainty dataset. Measures of uncertainty are derived via equation 4.13.



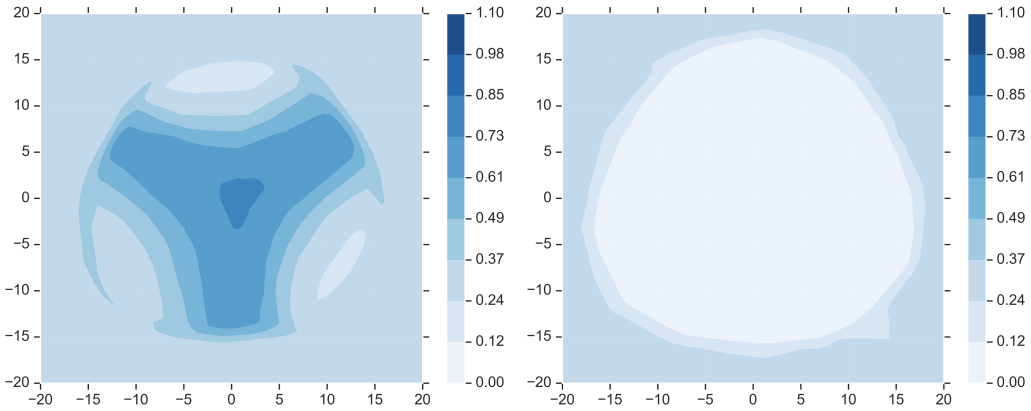
(a) Total Uncertainty - KL

(b) Total Uncertainty - reverse KL



(c) Expected Data Uncertainty - KL

(d) Expected Data Uncertainty - reverse KL

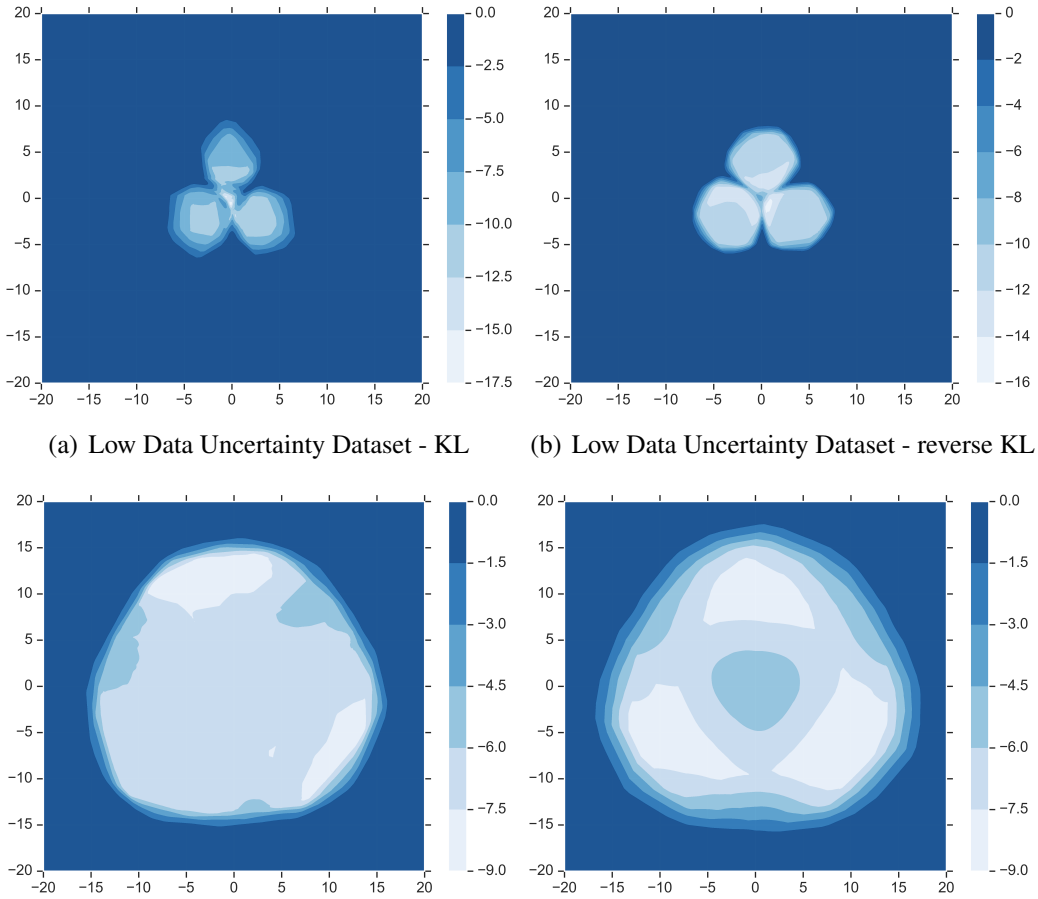


(e) Mutual Information - KL

(f) Mutual Information - reverse KL

Figure 4.5 Comparison of measures of uncertainty derived from Prior Networks trained with *forward* and *reverse* KL-divergence loss on the High Data Uncertainty dataset. Measures of uncertainty are derived via equation 4.13

uncertainty and mutual information, which is a measure of *knowledge uncertainty*. As was shown in equation 4.4, mutual information is the difference of *total uncertainty* and *expected data uncertainty*.



(a) Low Data Uncertainty Dataset - KL (b) Low Data Uncertainty Dataset - reverse KL

(c) High Data Uncertainty Dataset - KL (d) High Data Uncertainty Dataset - reverse KL

Figure 4.6 Comparison of Differential Entropy derived from Prior Networks trained with *forward* and *reverse* KL-divergence loss on the Low Data Uncertainty and High Data Uncertainty datasets. Differential entropy derived using equation 4.15.

Figures 4.4a and 4.4b show that Prior Networks trained using either the forward KL-divergence or reverse KL-divergence loss appropriately capture the *total uncertainty* of the LDU dataset. However, Prior Networks trained using forward KL-divergence do not fully capture *data uncertainty*, as figure 4.4c shows that *data uncertainty* is lower in the region where all three decision boundaries meet than along the decision boundaries where only two classes meet. As a result, the mutual information provided by a Prior Network trained with the forward KL-divergences is higher in-domain along the decision boundaries than out-of-domain. In contrast, figures 4.4d and 4.4f show that the measures of uncertainty

provided by a Prior Network trained using the reverse KL-divergence decompose correctly. *Data uncertainty* is highest along the decision boundaries and mutual information is 0 in in-domain, even along the decision boundaries.

These trends in the behaviours of uncertainty estimates are more apparent on the HDU dataset. By comparing figures 4.5a and 4.5b it is clear that a Prior Network trained using forward KL-divergence over-estimates *total uncertainty* in domain, as the *total uncertainty* is *equally* high along the decision boundaries, in the region of class overlap and out-of-domain. The Prior Network trained using reverse KL-divergence, on the other hand, yields a far more structured estimate of *total uncertainty*. Figure 4.5c shows that the *expected data uncertainty* is altogether incorrectly estimated by a Prior Network trained via forward KL-divergence. This causes mutual information, which is the difference of *total uncertainty* and *data uncertainty* to also behave incorrectly. On the other hand, the Prior Network trained via reverse KL-divergence yields correct decompositions of uncertainty.

Lastly, figure 4.6 depicts the behaviour of the differential entropy of Prior Network trained on the LDU and HDU datasets using both KL-divergence losses. Unlike the *total uncertainty*, *expected data uncertainty* and mutual information, it is less clear what is the desired behaviour of the differential entropy. Conceptually, it should be low in-domain and high out-of-distribution. Figures 4.6 shows that on both the LDU and HDU datasets both losses yield low differential entropy in-domain and high differential entropy out-of-distribution. However, the reverse KL-divergence seems to capture more of the structure of the dataset, which is especially evident in figure 4.6d, then the forward KL-divergence. However, in general both losses seem to yield an appropriate behaviour of differential entropy.

The experiments in this section support the analysis in the previous section and illustrate how the *reverse* KL-divergence is a generally more suitable optimization criterion than the *forward* KL-divergence, especially for datasets with a significant level of data uncertainty. However, it is important to keep in mind that the LDU and HDU are toy datasets where it is possible to obtain ideal out-of-distribution training data. However, the behaviours of Prior Networks trained using the forward and reverse KL-divergences losses may be different on real datasets.

4.3 Prior Networks for Regression

Having investigated Prior Networks for classification tasks in detail, we now consider Prior Networks for regression tasks. As discussed in section 4.1, Prior Networks yield an *explicit* conditional distribution $\mathtt{p}(\mathtt{p}_y|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})$ over output distribution, where $\mathtt{p}_y$ are the parameters

of the parametric forms of the output distributions. Here, output distributions $\mathrm{p}(\boldsymbol{y}|\mathrm{p}_y)$ over continuous variables $\boldsymbol{y} \in \mathcal{R}^K$ are considered.

The desired behaviour of a Prior Network for classification was depicted in figure 4.2. Unfortunately, it is difficult to visualize a distribution over the parameters of a probability density function which would convey the same degree of intuition. However, in chapter 3 figure 3.13 depicts the desired behaviour of an ensemble of 2D multivariate normal distributions. Specifically the ensemble should be consistent for in-domain inputs in regions of low/high *data uncertainty* and highly diverse both in the location of the mean and in the structure of the covariance for out-of-distribution inputs. Samples of continuous output distributions from a regression Prior Network should yield the same behaviour.

4.3.1 Parameterization and Uncertainty Measures

While there is a wide range of continuous output distributions, as discussed in sections 2.2.2 and 3.2.2, in this work only the multivariate normal (MVN) $\mathrm{p}(\boldsymbol{y}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ density function is considered. The corresponding Prior Network would parameterize a prior over the mean and covariance matrix $\mathrm{p}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}})$. This can be summarized in the general notation for Prior Networks:

$$\begin{aligned} \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\} &= \mathrm{p}_y \\ \mathrm{p}(\boldsymbol{y}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \mathrm{p}(\boldsymbol{y}|\mathrm{p}_y) \\ \mathrm{p}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}}) &= \mathrm{p}(\mathrm{p}_y|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}}) \end{aligned} \tag{4.34}$$

The conjugate prior distribution over multivariate normal distributions is the *Normal-inverse-Wishart* distribution $\mathcal{NW}^{-1}$, which can be parameterized using a neural network as follows:

$$\begin{aligned} \mathrm{p}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}}) &= \mathcal{NW}^{-1}(\boldsymbol{\mu}, \boldsymbol{\Sigma}; \hat{\boldsymbol{m}}, \hat{\boldsymbol{S}}, \hat{\kappa}, \hat{\nu}) \\ \{\hat{\boldsymbol{m}}, \hat{\boldsymbol{S}}, \hat{\kappa}, \hat{\nu}\} &= \boldsymbol{f}(\boldsymbol{x}^*; \hat{\boldsymbol{\theta}}),\ \hat{\kappa} > 0,\ \hat{\nu} > K - 1 \end{aligned} \tag{4.35}$$

where $\boldsymbol{m}$ and $\boldsymbol{S}$ are the *prior mean* and the positive-definite *prior scatter matrix*, while κ and ν are the strengths of belief in each prior, respectively. The parameters κ and ν are conceptually similar to *precision* of the Dirichlet distribution α_0. The Normal-inverse-Wishart is a compound distribution which decomposes into a product of a conditional normal distribution over the mean and an inverse-Wishart distribution over the covariance:

$$\begin{aligned} \mathcal{NW}^{-1}(\boldsymbol{\mu}, \boldsymbol{\Sigma}; \boldsymbol{m}, \boldsymbol{S}, \kappa, \nu) &= \mathcal{N}(\boldsymbol{\mu}; \boldsymbol{m}, \frac{1}{\kappa}\boldsymbol{\Sigma}) \cdot \mathcal{W}^{-1}(\boldsymbol{\Sigma}; \boldsymbol{S}, \nu) \\ &= \mathrm{p}(\boldsymbol{\mu}|\boldsymbol{\Sigma}) \cdot \mathrm{p}(\boldsymbol{\Sigma}) \end{aligned} \tag{4.36}$$

The inverse-Wishart distribution $\mathcal{W}^{-1}$ is a distribution over positive-definite symmetric matrices $\boldsymbol{\Sigma}$ of size $K \times K$ defined as follows:

$$\mathcal{W}^{-1}(\boldsymbol{\Sigma}; \boldsymbol{S}, \nu) = \frac{|\boldsymbol{S}|^{\frac{\nu}{2}}}{2^{\frac{\nu K}{2}} \Gamma_K(\frac{\nu}{2})} |\boldsymbol{\Sigma}|^{\frac{-(\nu+K+1)}{2}} e^{-\frac{1}{2}\mathbf{tr}(\boldsymbol{S}\boldsymbol{\Sigma}^{-1})}, \ \nu \geq K-1 \tag{4.37}$$

where $\Gamma_K(\cdot)$ is the *multivariate gamma function* and K is the dimensionality of $\boldsymbol{y}$. The decomposition of the Normal-inverse-Wishart prior $\mathrm{p}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ into a conditional prior over the mean $\mathrm{p}(\boldsymbol{\mu}|\boldsymbol{\Sigma})$ and a prior over the covariance $\mathrm{p}(\boldsymbol{\Sigma})$ illustrates how *knowledge uncertainty* in the prediction $\boldsymbol{\mu}$ and the *data uncertainty*, described by $\boldsymbol{\Sigma}$, can be separately discussed for regression.

Given a Prior Network which parameterizes a Normal-inverse-Wishart distribution, the expected predictive posterior distribution will be the multivariate $\mathcal{T}$ distribution:

$$\begin{aligned} \mathrm{p}(\boldsymbol{y}|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}}) &= \mathbb{E}_{\mathrm{p}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}})} \big[\mathrm{p}(\boldsymbol{y}|\boldsymbol{\mu}, \boldsymbol{\Sigma})\big] \\ &= \mathcal{T}(\boldsymbol{y}|\hat{\boldsymbol{m}}, \frac{\hat{\kappa}+1}{\hat{\kappa}(\hat{\nu}-K+1)}\hat{\boldsymbol{S}}, \hat{\nu}-K+1), \ \hat{\nu} \geq K-1, \ \hat{\kappa} > 0 \end{aligned} \tag{4.38}$$

The $\mathcal{T}$ distribution is heavy-tailed generalization of the multivariate normal distribution defined as:

$$\mathcal{T}(\boldsymbol{y}|\boldsymbol{\mu}, \boldsymbol{S}, \nu) = \frac{\Gamma(\frac{\nu+K}{2})}{\Gamma(\frac{\nu}{2})\nu^{\frac{K}{2}}\pi^{\frac{K}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \Big(1 + \frac{1}{\nu}(\boldsymbol{y}-\boldsymbol{\mu})^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{y}-\boldsymbol{\mu})\Big)^{-\frac{(\nu+K)}{2}}, \ \nu \geq 0 \tag{4.39}$$

where ν is the number of degrees of freedom. In the limit, as $\nu \to \infty$, the $\mathcal{T}$ distribution converges to a normal distribution. The mean, mode and median of the $\mathcal{T}$ distribution are all equal to $\boldsymbol{\mu}$, as the $\mathcal{T}$ distribution is symmetric. However, the mean is only defined when $\nu > 1$ and the variance is defined only when $\nu > 2$. This means that the predictive posterior of the Prior Network given in equation 4.38 only has a defined mean and variance when $\hat{\nu} > K+1$.

Given a Normal-inverse-Wishart Prior Network it is possible to compute closed-form expression for all measures of uncertainty discussed in section 4.1, including the differential entropy of the expected posterior $\mathcal{H}\big[\mathrm{p}(\boldsymbol{y}|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}})\big]$, the differential entropy of the Prior Network $\mathcal{H}[\mathrm{p}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}})]$, the mutual information $\mathcal{I}[\boldsymbol{y}, (\boldsymbol{\mu}, \boldsymbol{\Sigma})|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}}]$ and the expected pairwise KL-divergence $\mathcal{K}[\mathrm{p}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\boldsymbol{x}^*; \hat{\boldsymbol{\theta}})]$. However, these expressions are omitted and instead provided in appendix A, as they are quite complex and give little additional insight.

4.3.2 Training Criteria

Having discussed how to construct Prior Networks for regression, we now discuss how they can be trained. As discussed in section 4.1, Prior Networks are trained using multi-task training where an in-domain loss $\mathcal{L}_{in}(\boldsymbol{\theta}, \mathcal{D}_{tr})$ and an out-of-distribution loss $\mathcal{L}_{out}(\boldsymbol{\theta}, \mathcal{D}_{out})$ are jointly minimized:

$$\mathcal{L}(\boldsymbol{\theta}, \mathcal{D}) = \mathcal{L}_{in}(\boldsymbol{\theta}, \mathcal{D}_{tr}) + \gamma \cdot \mathcal{L}_{out}(\boldsymbol{\theta}, \mathcal{D}_{out}) \tag{4.40}$$

Just as in the case of classification tasks, it is necessary to have the Prior Network yield a high-entropy prior distribution over output distributions for out-of-distribution inputs and a low-entropy prior distribution over output distributions in-domain. Specifically, samples of continuous output distributions from a regression Prior Network should be consistent for in-domain inputs in regions of low/high *data uncertainty* and highly diverse both in the location of the mean and in the structure of the *data uncertainty* for out-of-distribution inputs.

First, let's consider how to specify the loss $\mathcal{L}_{in}(\boldsymbol{\theta}, \mathcal{D}_{tr})$ to train the model to yield the desired in-domain behaviour. Both the Normal and inverse-Wishart distribution are conjugate priors and a members of the exponential family of distribution. Thus, by analogy with Prior Networks for classification, the consistent choice of in-domain loss function is the *reverse* KL-divergence between the model and a target Normal-inverse-Wishart distribution $\mathrm{p}(\mathrm{p}_y; \tilde{\boldsymbol{m}}, \tilde{\boldsymbol{S}}, \tilde{\kappa}, \tilde{\nu})$:

$$\begin{aligned}\mathcal{L}_{in}^{RKL}(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}) &= \mathtt{KL}[\mathrm{p}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\boldsymbol{x}; \boldsymbol{\theta})||\mathrm{p}(\boldsymbol{\mu}, \boldsymbol{\Sigma}; \tilde{\boldsymbol{m}}, \tilde{\boldsymbol{S}}, \tilde{\kappa}, \tilde{\nu})] \\ &= \underbrace{-\mathbb{E}_{\mathrm{p}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\boldsymbol{x}; \boldsymbol{\theta})}[\ln \mathrm{p}(\boldsymbol{\mu}, \boldsymbol{\Sigma}; \tilde{\boldsymbol{m}}, \tilde{\boldsymbol{S}}, \tilde{\kappa}, \tilde{\nu})]}_{Reverse\ Cross\ Entropy} - \underbrace{\mathcal{H}[\mathrm{p}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\boldsymbol{x}; \boldsymbol{\theta})]}_{Differential\ Entropy}\end{aligned} \tag{4.41}$$

This loss decomposes into the reverse cross-entropy and the negative differential entropy. As in the case for classification, the negative differential entropy can be seen as a regularizer which prevents the distribution over normal distributions from becoming too sharp. Let's

consider the reverse cross-entropy term in greater detail:

$$\begin{aligned}
\mathcal{L}_{in}^{RCE}(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta}) =& -\mathbb{E}_{\mathtt{p}(\boldsymbol{\mu},\boldsymbol{\Sigma}|\boldsymbol{x};\boldsymbol{\theta})}\big[\ln \mathtt{p}(\boldsymbol{\mu}, \boldsymbol{\Sigma}; \tilde{\boldsymbol{m}}, \bar{\boldsymbol{S}}, \bar{\kappa}, \bar{\nu})\big] \\
=& -\mathbb{E}_{\mathtt{p}(\boldsymbol{\mu},\boldsymbol{\Sigma}|\boldsymbol{x};\boldsymbol{\theta})}\Big[\frac{-(\tilde{\nu}+K+2)}{2}\ln|\boldsymbol{\Sigma}| - \frac{1}{2}\mathtt{tr}(\tilde{\boldsymbol{S}}\boldsymbol{\Sigma}^{-1}) \\
& - \frac{\tilde{\kappa}}{2}(\boldsymbol{\mu}-\tilde{\boldsymbol{m}})^{\mathtt{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}-\tilde{\boldsymbol{m}})\Big] \\
=& \frac{(\tilde{\nu}+K+2)}{2}\Big(\ln|\hat{\boldsymbol{S}}| - K\ln 2 - \sum_{c=1}^{K}\psi\big(\frac{\hat{\nu}-K+c}{2}\big)\Big) + \frac{K\tilde{\kappa}}{2\hat{\kappa}} \\
& + \frac{1}{2}\mathtt{tr}\Big(\hat{\nu}\hat{\boldsymbol{S}}^{-1}\big(\tilde{\boldsymbol{S}} + \tilde{\kappa}(\hat{\boldsymbol{m}}-\tilde{\boldsymbol{m}})(\hat{\boldsymbol{m}}-\tilde{\boldsymbol{m}})^{\mathtt{T}}\big)\Big) + \ln\tilde{\boldsymbol{Z}}
\end{aligned} \tag{4.42}$$

where the normalization constant $\tilde{\boldsymbol{Z}}$ is:

$$\ln\tilde{\boldsymbol{Z}} = \frac{\tilde{\nu}K}{2}\ln 2 + \ln\Gamma_K(\frac{\tilde{\nu}}{2}) + \frac{D}{2}\ln(\frac{2\pi}{\tilde{\kappa}}) - \frac{\nu_0}{2}\ln|\tilde{\boldsymbol{S}}| \tag{4.43}$$

Where the target parameters are a function of both $\boldsymbol{x}$ and $\boldsymbol{y}$, as defined below:

$$\begin{aligned}
\tilde{\boldsymbol{m}} &= \frac{\kappa_0}{\kappa_0+\beta}\boldsymbol{m}_0 + \frac{\beta}{\kappa_0+\beta}\boldsymbol{y} \\
\tilde{\boldsymbol{S}} &= \nu_0\boldsymbol{\Sigma}_0 + \frac{\beta\kappa_0}{\kappa_0+\beta}(\boldsymbol{y}\boldsymbol{y}^{\mathtt{T}} + \boldsymbol{m}_0\boldsymbol{m}_0^{\mathtt{T}} - 2\boldsymbol{m}_0\boldsymbol{y}^{\mathtt{T}}) \\
\tilde{\kappa} &= \kappa_0+\beta, \quad \tilde{\nu} = \nu_0+\beta \\
\{\boldsymbol{m}_0, \boldsymbol{\Sigma}_0\} &= \boldsymbol{f}(\boldsymbol{x}; \tilde{\boldsymbol{\theta}})
\end{aligned} \tag{4.44}$$

Here $\boldsymbol{m}_0$ and $\boldsymbol{\Sigma}_0$ can be obtained via a *teacher* density network $\mathtt{p}(\boldsymbol{y}|\boldsymbol{x};\tilde{\boldsymbol{\theta}})$ trained via maximum likelihood. As defined above, the parameters make use of the conjugacy property of the Normal-inverse-Wishart distribution and represent the parameters of a Normmal-inverse-Wishart posterior after observing the training data. The parameter β represent how strongly we believe in the observations and κ_0 and ν_0 represent how strongly we believe in the prior estimates of the mean and scatter matrix, respectively. If the density network is trained on the same dataset as the regression Prior Network, then conceptually there is not much point in using prior maximum likelihood estimates for the mean and scatter matrix. In contrast, if a model trained on one dataset is being adapted to a new dataset, it is possible to control the degree to which the model is biased to the new data via β, κ_0 and ν_0. However, as the latter situation is not considered in this thesis, the parameters of the target

Normal-inverse-Wishart can be set to:

$$\begin{aligned} \tilde{\boldsymbol{m}} = \boldsymbol{y}, \quad \tilde{\boldsymbol{S}} = \mathbf{0} \\ \tilde{\kappa} = \beta, \quad \tilde{\nu} = \beta \end{aligned} \tag{4.45}$$

This greatly simplifies the expression for the loss. Taking the expectation of the reverse cross entropy with respect to the empirical distribution $\hat{\mathtt{p}}(\boldsymbol{x}, \boldsymbol{y})$ yields:

$$\begin{aligned} \mathcal{L}_{in}^{RCE}(\boldsymbol{\theta}, \mathcal{D}_{tr}) &= \mathbb{E}_{\hat{\mathtt{p}}(\boldsymbol{x},\boldsymbol{y})}\Big[\mathcal{L}_{in}^{RCE}(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{\theta})\Big] \\ &= \mathbb{E}_{\hat{\mathtt{p}}(\boldsymbol{x})}\Big[\frac{(\beta + K + 2)}{2}\Big(\ln|\hat{\boldsymbol{S}}| - \sum_{c=1}^{K}\psi\big(\frac{\hat{\nu} - K + c}{2}\big)\Big) + \frac{K\beta}{2\hat{\kappa}} \\ &\quad + \frac{\beta}{2}\mathtt{tr}\Big(\hat{\nu}\hat{\boldsymbol{S}}^{-1}\big(\boldsymbol{\Sigma}_{tr} + (\hat{\boldsymbol{m}} - \boldsymbol{\mu}_{tr})(\hat{\boldsymbol{m}} - \boldsymbol{\mu}_{tr})^{\mathtt{T}}\big)\Big)\Big] \end{aligned} \tag{4.46}$$

Estimates *data uncertainty* will be naturally captured via the last term in equation 4.46, while the first term regularizes the covariance matrix. In general, this expression is an upper bound to the *forward* KL-divergence between the normal distribution and the underlying distribution of the training data, with additional terms that drive $\hat{\kappa}$ and $\hat{\nu}$ to be large. This shows that reverse KL-divergence is an appropriate loss function for Prior Network parameterizing the Normal-inverse-Wishart distribution.

Curiously, the expression for the *reverse* KL divergence between Normal-inverse-Wishart distributions contain within itself the expression for the upper bound to the *forward* cross-entropy between normal distributions. As was shown in section 4.2.2, the expression *reverse* KL divergence between Dirichlet distributions contains within it the expression for the *forward* KL-divergence between discrete distributions. It is possible this 'anti-symmetry' is a general property of α-divergences between members of the exponential family of distributions and should be explored in future work.

Similarly, a Prior Network can be trained to yield a high-entropy distribution for out-of-domain inputs by minimizing the reverse KL-divergence between the model and a high-entropy target Normal-inverse-Wishart distributions:

$$\mathcal{L}_{out}^{RKL}(\boldsymbol{\theta}, \mathcal{D}_{out}) = \mathbb{E}_{\hat{\mathtt{p}}(\boldsymbol{x})}\big[\mathtt{KL}[\mathtt{p}(\mathtt{p}_y|\boldsymbol{x};\boldsymbol{\theta})||\mathtt{p}(\mathtt{p}_y;\tilde{\boldsymbol{m}},\tilde{\boldsymbol{S}},\tilde{\kappa},\tilde{\nu})]\big] \tag{4.47}$$

Conceptually, choosing an out-of-distribution target distribution is similar to choosing the mean and variance to which a Gaussian Process reverts far away from training data. One possible choice of target out-of-domain distribution is a *uninformative*, improper Normal-

inverse-Wishart distribution obtained by setting the parameters as follows:

$$\begin{aligned} \tilde{\boldsymbol{m}} &= \mathbf{0}, \quad \tilde{\kappa} = 1e-6 \\ \tilde{\boldsymbol{S}} &= \boldsymbol{I}, \quad \tilde{\nu} = 1e-6 \end{aligned} \tag{4.48}$$

Under an improper, uninformative prior all possible output distributions are, in some sense, 'equally likely', which corresponds to a situation where the model doesn't understand the data. However, it may be better to choose a *weakly informative* prior distribution over multivariate normal distributions, where $\tilde{\boldsymbol{m}}$ and $\tilde{\boldsymbol{S}}$ are set the marginal mean and scatter matrix of the targets, respectively, and where ν and κ are set to small values [92]:

$$\begin{aligned} \tilde{\boldsymbol{m}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{y}^{(i)}, \quad \tilde{\boldsymbol{S}} = \sum_{i=1}^{N} (\boldsymbol{y}^{(i)} - \tilde{\boldsymbol{m}})(\boldsymbol{y}^{(i)} - \tilde{\boldsymbol{m}})^{\mathsf{T}} \\ \tilde{\kappa} = 1e-3, \quad \tilde{\nu} = K + 2 \end{aligned} \tag{4.49}$$

Thus, the model regresses to the marginal distribution over $\boldsymbol{y}$ for out-of-domain inputs, but indicates a large degree of *knowledge uncertainty* via low values of $\hat{\kappa}$ and $\hat{\nu}$. The weight γ would control the boundary of the in-domain and out-of-domain regions by appropriately smoothing the parameters of the Normal-inverse-Wishart distribution between the two regions.

4.4 Chapter Summary

This chapter introduced a new class of models for modelling uncertainty, called *Prior Networks*. In section 4.1, by drawing analogy with ensembles, which can be seen as samples from an *implicit* distribution over output distributions, it was proposed to *explicitly* parameterize a distribution over output distributions using a single neural network. These models, called Prior Networks, effectively emulate an ensemble of models using a single neural network, combining the elegant theoretical properties of ensembles with the practical advantages of single model approaches. Specifically, by yielding the same measures of uncertainty as ensembles, Prior Networks are able to decompose the sources of uncertainty within a single coherent probabilistic framework. This is important for tasks where it is necessary to know the source of uncertainty, such as active learning. Furthermore, Prior Networks provide a clear link between single-model and ensemble approaches - *both* approaches capture uncertainty via a *conditional distribution over output distributions*. The difference between the two approaches is in the way the behaviour of the distribution is controlled - either via an implicit or explicit prior $\mathrm{p}(\mathcal{M})$ and approximate inference scheme or via the *training data* $\mathcal{D}$.

In section 4.2 the construction of Prior Networks for classification by parameterizing the Dirichlet distribution was investigated and closed form expressions for measures of uncertainty were provided. In section 4.2.2 the theoretical properties of several loss functions were investigated. It was shown that *reverse KL-divergence* loss is theoretically able to train a Prior Network to yield the desired set of in-domain and out-of-distribution behaviour described in figure 4.2. In section 4.2.3 the forward and reverse KL divergences losses were used to train toy Prior Networks on the Low Data Uncertainty and High Data Uncertainty datasets introduced in the previous chapter. The experiments confirmed the theoretical properties of both losses and showed that the reverse KL-divergence loss empirically yields the desired set of behaviours.

Section 4.3 investigated the construction of Prior Networks for regression by parameterizing the Normal-inverse-Wishart distribution, the conjugate prior to the multivariate Normal distribution. By analogy with classification Prior Networks, section 4.3.2 investigated the training of regression Prior Networks via the reverse KL-divergence loss and discussed appropriate construction of target in-domain and out-of-domain Normal-inverse-Wishart distributions.

Chapter 5

Experimental Evaluation of Prior Networks

The previous chapter introduced a new class of models called *Prior Networks* which combined aspects of single model and ensemble approaches to uncertainty estimation. Prior Networks allow both *data* and *knowledge uncertainty* to be modeled within a consistent probabilistic framework using a single neural network. The previous chapter examined a range of different loss functions for Prior Networks and determined that the *reverse* KL-divergence between the Prior Network and a target Dirichlet distribution is an appropriate loss. It was shown that by using this loss, Prior Networks can be successfully trained on the artificial datasets introduced in chapter 3 to yield the desired set of behaviours described in section 4.1, providing interpretable measures of uncertainty.

The current chapter aims to evaluate the performance of Prior Networks on a range of increasingly complex image classification datasets, specifically the MNIST [70], SVHN [41] and CIFAR-10 [65] datasets, and compare them to previous methods for uncertainty estimation. Prior Networks will be evaluated on two related practical applications of uncertainty estimation - misclassification detection and out-of-distribution (OOD) sample detection. Both of these tasks are binary classification tasks based on measures of uncertainty - if the uncertainty is higher than a threshold, then the input is considered a misclassification or out-of-distribution sample.

In these experiments Prior Networks were compared to a standard DNN, an ensemble derived via Monte-Carlo Dropout (MCDP) [35, 36] from a standard DNN and an explicit ensemble of standard DNNs obtained by using different random initializations [68]. Prior Networks are not compared to single-model approaches described in section 3.3.1 because they are a generalization of these approaches, as discussed in the previous chapter.

The structure of the current chapter is as follows: section 5.1 describes the datasets and details the construction of the models used in the following experiments; the evaluation of the quality of uncertainty estimates and the metrics of performance on misclassification and OOD sample detection tasks are described in section 5.2; missclassification and OOD sample detection experiments are described in sections 5.3 and 5.4, respectively.

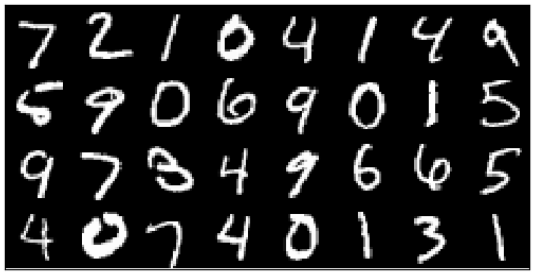
5.1 Datasets and Experimental Setup

In this chapter Prior Networks are trained on the MNIST [70], SVHN [41] and CIFAR-10 [65] datasets, described in table 5.1. These datasets are consider 'in-domain' for the models trained on them. MNIST is a dataset of 28×28 images of black-and-white hand-written digits; SVHN is a color dataset of 32×32 images of house numbers in Google Street-view, where the target class is the middle digit; finally, CIFAR-10 is a color dataset of 32×32 images of 10 real object categories. These datasets are increasingly challenging image classification tasks as the degree of variation of each class increases, which can be seen from samples of each dataset shown in figure 5.1. All datasets have roughly the same number of training examples (between 50K and 70K), though the SVHN dataset has an additional set of 531K 'extra' training images which is not used in the current experiments.

Domain	Dataset	Train	Valid	Test	Extra	Classes
In-Domain	MNIST	55000	5000	10000	-	10
	SVHN	73257	-	26032	531131	
	CIFAR-10	50000	-	10000	-	
Out-of-Domain	Semeion	1593				10
	Omniglot	32460				1623
	LSUN	-	-	10000	-	10
	CIFAR-100	50000	-	10000	-	100
	TinyImagenet	100000	10000	10000	-	200

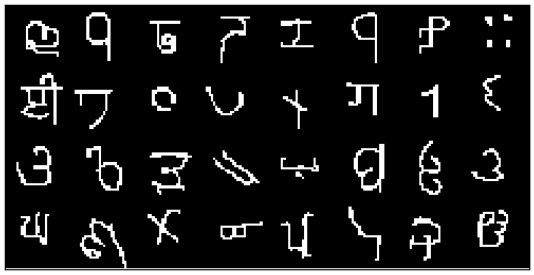
Table 5.1 Description of in-domain and out-of-domain datasets in terms of number of images and classes.

In addition to the *in-domain* datasets described above, a series of *out-of-distribution* datasets are used for training Prior Networks and as evaluation data for out-of-distribution detection experiments. Images from these datasets are shown in figure 5.1 and described in table 5.1. The Omniglot [67] and Semeion [16] datasets are used as OOD data for Prior Networks trained on MNIST. Like MNIST, Semeion is a dataset of black-and-white



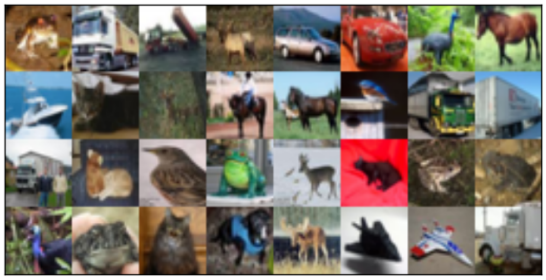
(a) MNIST

(b) Semeion

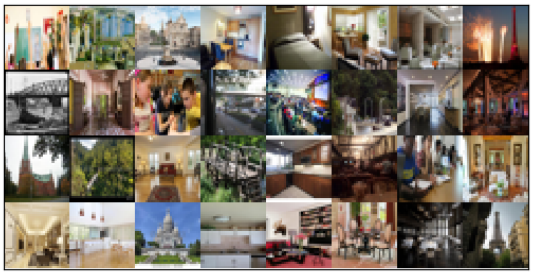


(c) Omniglot

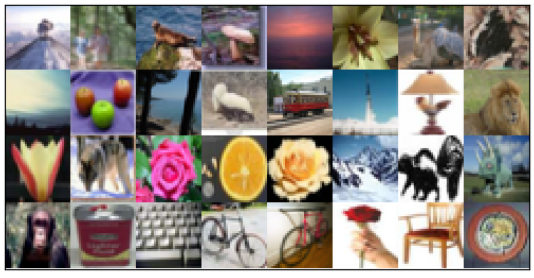
(d) SVHN



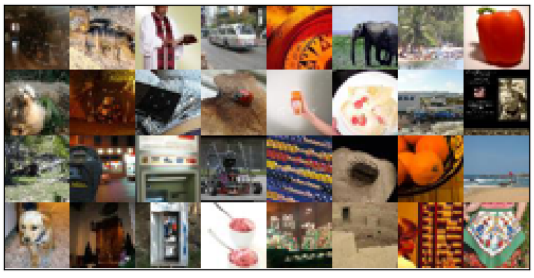
(e) CIFAR-10



(f) LSUN



(g) CIFAR-100



(h) TinyImagenet

Figure 5.1 Samples of images from all datasets

handwritten digits, but with a different style. Omniglot is a dataset of 1623 unique 92×92 black-and-white handwritten characters from 50 different alphabets. Omniglot images are resized down to 28x28 for comparison to MNIST. This corresponds to images which are in a similar 'style' to MNIST, but which describe completely different classes. A more detailed breakdown of Omniglot into subsets is described in table 5.2. The Omniglot BG set contains different alphabets and therefore characters from the EVAL set. BGS1 and BGS2 are subsets of the large BG set. The LSUN [135], CIFAR-100 [65] and TinyImagenet [24] datasets are used as OOD data for Prior Networks trained on SVHN and CIFAR-10. LSUN is a 32×32 color dataset of 10 different scene categories, such as bedrooms and so on. CIFAR-100 is a 32×32 color dataset similar to CIFAR-10, but which describes 100 object categories which are not present in CIFAR-10. Finally, TinyImagenet is a 200-class subset of the ImageNet [27] dataset of real objects. TinyImagenet was resized down to 32×32 from 64×64. For all datasets the input features were re-scaled the range -1 to 1 from the range 0 to 255.

Subset	Alphabets	Size	Classes
BGS1	5	2720	136
BGS2	5	3120	156
BG	30	19280	964
EVAL	20	13180	659

Table 5.2 Detailed description of Omniglot dataset in terms of number of alphabets, images and classes. BGS1 and BGS2 are non-overlapping subsets of BG. BG and EVAL are also non-overlapping datasets.

5.1.1 Model architecture and training

All models considered in this chapter were implemented in Tensorflow [4] using variants on the VGG [113] architecture for image classification. DNN models were trained using the negative log-likelihood loss. Prior Networks were trained using both the KL-divergence (PN-KL) and reverse KL-divergence (PN-RKL) losses introduced in section 4.2.2 to compare their behaviour on more challenging datasets. Identical target concentration parameters $\boldsymbol{\beta}^{(c)}$ were used for both the forward and reverse KL-divergence losses. All models were trained using the Adam [62] optimizer, described in section 2.3, with a 1-cycle learning rate policy and dropout regularization. In additional, data augmentation was done when training models on the CIFAR-10 dataset via random left-right flips, random shifts up to ± 4 pixels and random rotations by up to $\pm$ 15 degrees. The details of the training configurations for all models and each dataset can be found in table 5.3. 10 models of each type were trained

Training Dataset	Model	General				Prior Network Only		
		η_0	Epochs	Cycle len.	Dropout	γ	β	OOD data
MNIST	DNN	1e-3	20	10	0.5			-
	PN-KL				0.9	1.0	1e3	MNIST FA
	PN-RKL	1e-3	20	10	0.9	10.0	1e3	MNIST FA Omni-BGS1 Omni-BGS1+2 Omni-BG
SVHN	DNN	1e-3	40	30	0.5			-
	PN-KL	5e-4			0.7	1.0	1e3	CIFAR-10
	PN-RKL	5e-4			0.7	10.0		
CIFAR-10	DNN	1e-3	45	30	0.5			-
	PN-KL	5e-4			0.7	1.0	1e2	CIFAR-100
	PN-RKL	5e-4			0.7	10.0		

Table 5.3 Training Configurations. η_0 is the initial learning rate, γ is the out-of-distribution loss weight and β is the concentration of the target class. The batch size for all models was 128. Dropout rate is quoted in terms of probability of *not* dropping out a unit.

starting from different random seeds. The 10 DNN models were evaluated both individually (DNN) and as an explicit ensemble of models (ENS). Additionally, each of the DNN models was also used to construct an ensemble via Monte-Carlo dropout (MCDP).

5.1.2 Out-of-distribution training data

As discussed in the previous chapter, out-of-distribution training data is necessary to train Prior Networks so that they yield high-entropy distributions over distributions out-of-domain. In this chapter Prior Networks trained on SVHN and CIFAR-10 use the CIFAR-10 and CIFAR-100 training datasets, described in table 5.1 as OOD training data, respectively. This OOD training data is augmented by random left-right and up-down flips, random shifts by up to ± 4 pixels and random rotations by up to $\pm$ 15 degrees. On the MNIST dataset, however, generation of OOD training data was investigated. Priors Network trained on MNIST use out-of-distribution training data synthesized using Factor Analysis [92], which is a linear generative model.

A standard factor analysis model is described below:

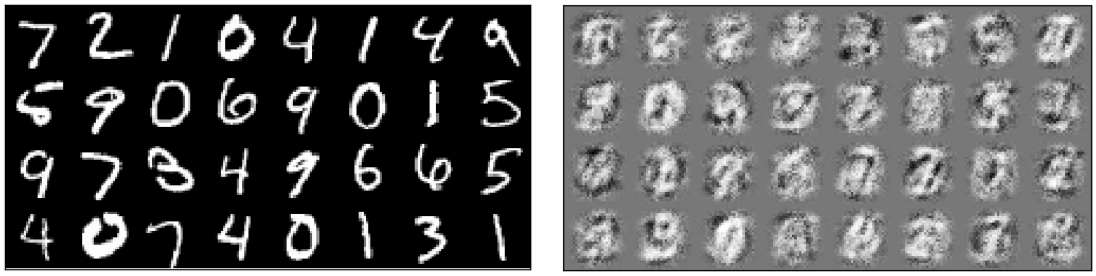
$$\begin{aligned} \boldsymbol{z} &\sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \\ \boldsymbol{x} &\sim \mathcal{N}(\boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu}, \boldsymbol{\Sigma}) \end{aligned} \tag{5.1}$$

where $\boldsymbol{W}$ is the loading matrix, $\boldsymbol{\mu}$ is the global mean, $\boldsymbol{\Sigma}$ is a *diagonal* covariance matrix and $\boldsymbol{z}$ is a latent vector sampled from an isotropic standard normal distribution. The idea is to model the covariances via $\boldsymbol{z}$ and the loading matrix. Here, a factor analysis model with a 50-dimensional latent space was trained on the MNIST data.

As described in section 3.3, out-of-distribution training data should lie close to the boundary of the in-domain region so that a model trained in a multi-task fashion may learn a tight decision boundary to separate the in-domain region from everything else. Thus, to push the factor analysis model to produce data at the 'boundary' of the in-domain region the variance on the latent distribution and the global covariance matrix were increased by a factor λ:

$$\begin{aligned} \boldsymbol{z} &\sim \mathcal{N}(\boldsymbol{0}, \lambda \cdot \boldsymbol{I}) \\ \boldsymbol{x} &\sim \mathcal{N}(\boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu}, \lambda \cdot \boldsymbol{\Sigma}) \end{aligned} \tag{5.2}$$

The synthetic data generated using this factor analysis model for a $\lambda = 3$ is visualized alongside MNIST data in figure 5.2. Clearly, this data is quite different from the MNIST data, though it is possible to see shapes reminiscent of the MNIST data. As was previous



(a) MNIST (b) Samples from Factor Analysis model

Figure 5.2 Comparison of MNIST and Factor Analysis generated images. Factor Analysis images were samples using equation 5.2 with $\lambda = 3$.

stated in chapter 4, a proper, in-depth investigation of out-of-distribution data generation using state-of-the-art generative models is beyond the scope of this thesis. However, using the Omniglot BGS1, BGS2 and BG datasets, described in table 5.2, as OOD training data for Prior Networks is explored.

5.2 Evaluation Metrics

In this chapter two practical applications of uncertainty estimation are considered - misclassification detection and out-of-distribution sample detection. Both can be seen as an outlier detection task based on measures of uncertainty, where misclassifications are one form of outlier and out-of-distribution inputs are another form of outlier. Both of these tasks can be formulated as threshold-based binary classification [49]. Here a detector $\mathcal{I}_T(\boldsymbol{x})$ assigns the label 1 (uncertain prediction) if an uncertainty measure $\mathcal{H}(\boldsymbol{x})$ is above a threshold T, and label 0 (confident prediction) otherwise. This uncertainty measure can be any of the measures discussed in chapters 3 and 4.

$$\mathcal{I}_T(\boldsymbol{x}) = \begin{cases} 1, \ \mathcal{H}(\boldsymbol{x}) > T \\ 0, \ \mathcal{H}(\boldsymbol{x}) \leq T \end{cases} \tag{5.3}$$

Given a set of true positive examples $\mathcal{D}_{\mathtt{p}} = \{\boldsymbol{x}_p^{(i)}\}_{i=1}^{N_p}$ and a set of true negative examples $\mathcal{D}_{\mathtt{n}} = \{\boldsymbol{x}_n^{(j)}\}_{j=1}^{N_n}$ the performance of such a detection scheme can be evaluated at a particular threshold value T using the *true positive rate* $t_p(T)$ and the *false positive rate* $f_p(T)$:

$$t_p(T) = \frac{1}{N_p} \sum_{i=1}^{N_p} \mathcal{I}_T(\boldsymbol{x}_p^{(i)}) \qquad f_p(T) = \frac{1}{N_n} \sum_{j=1}^{N_n} \mathcal{I}_T(\boldsymbol{x}_n^{(j)}) \tag{5.4}$$

The range of trade-offs between the true positive and the false positive rates can be visualized using a Receiver-Operating-Characteristic (ROC) and the quality of the possible trade-offs can be summarized using the area under the ROC curve (AUROC) [92]. If there are significantly more negatives than positives, however, this measure will over-estimate the performance of the model and yield a high AUROC value [92]. In this situation it is better to calculate the *precision* and *recall* of this detection scheme at every threshold value and plot them against each other on a Precision-Recall (PR) curve [92]. The recall $R(T)$ is equal to the true positive rate $t_p(T)$, while precision measures the number of true positives among all samples labelled as positive:

$$P(T) = \frac{\sum_{i=1}^{N_p} \mathcal{I}_T(\boldsymbol{x}_p^{(i)})}{\sum_{i=1}^{N_p} \mathcal{I}_T(\boldsymbol{x}_p^{(i)}) + \sum_{j=1}^{N_n} \mathcal{I}_T(\boldsymbol{x}_n^{(j)})} \qquad R(T) = \frac{1}{N_p} \sum_{i=1}^{N_p} \mathcal{I}_T(\boldsymbol{x}_p^{(i)}) \tag{5.5}$$

The quality of the trade-offs can again be summarized via the area under the PR curve (AUPR). For both the ROC and the PR curves an ideal detection scheme will achieve an AUC of 100%. A completely random detection scheme will have an AUROC of 50% and

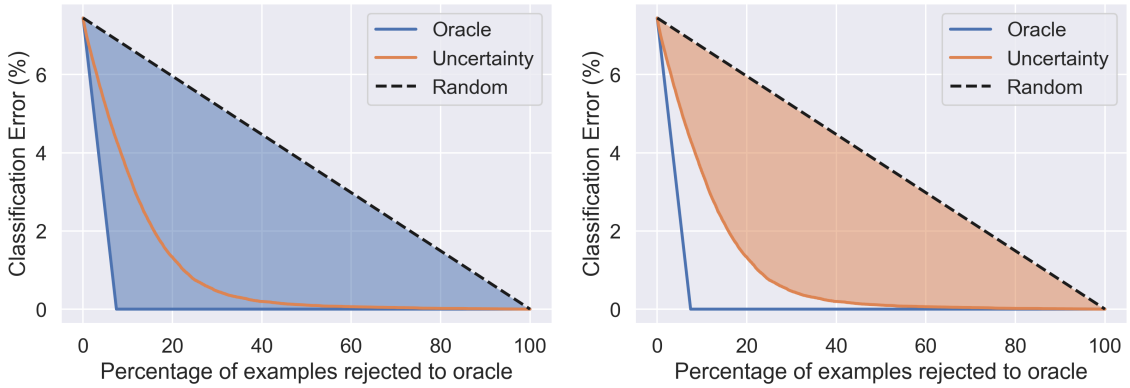
the AUPR will be the ratio of the number of positive examples to the total size of the dataset (positive and negative) [92].

A crucial difference between the ROC curve and the PR curve is that the ROC curve is not sensitive to the balance of positive and negative examples in the dataset and returns overall classification performance, unlike the PR curve, which will be more sensitive to the performance on the positive class [92]. If the number of positives and negatives is the same, the ROC and PR curves yield the same information. However, if the number of positives and negatives is heavily skewed, the AUPR will be a more informative metric of performance. Alternative metrics, like F-scores [92], can also be used to summarize detection performance, but they report performance only at a particular threshold value, while AUROC and AUPR summarize the entire range of possible operating points. In this work the positive class is chosen to correspond to either misclassifications or OOD inputs, depending on the task. This is because on the MNIST, SVHN and CIFAR-10 datasets examined in this work a typical DNN classifier will have a lower number of misclassifications than correct predictions, and we are are specifically interested in how well measures of uncertainty can detect misclassifications.

Consider the task of misclassification detection - ideally we would like to detect all of the inputs which the model has misclassified based on a measure of uncertainty. Then, the model can either choose to not provide any prediction for these inputs, or they can be passed over or 'rejected' to an oracle (ie: human) to obtain the correct prediction. The latter process can be visualized using a *rejection curve* depicted in figure 5.3, where the predictions of the model are replaced with predictions provided by an oracle in some particular order based on estimates of uncertainty. If the estimates of uncertainty are 'useless', then, in expectation, the rejection curve would be a straight line from base error rate to the lower right corner. However, if the estimates of uncertainty are 'perfect' and always bigger for a misclassification than for a correct classification, then they would produce the 'oracle' rejection curve. A rejection curve produced by estimates of uncertainty which are not perfect, but still informative, will sit between the 'random' and 'oracle' curves. The quality of the rejection curve can be assessed by considering the *ratio* of the area between the 'uncertainty' and 'random' curves $AR_{\texttt{uns}}$ (orange in figure 5.3) and the area between the 'oracle' and 'random' curves $AR_{\texttt{orc}}$ (blue in figure 5.3). This yields the *rejection area ratio* RR:

$$RR = \frac{AR_{\texttt{uns}}}{AR_{\texttt{orc}}} \tag{5.6}$$

A rejection area ratio of 1.0 indicates optimal rejection, a ratio of 0.0 indicates 'random' rejection. A negative rejection ratio indicates that the estimates of uncertainty are 'perverse'



(a) Shaded area is $AR_{\texttt{orc}}$.

(b) Shaded area is $AR_{\texttt{uns}}$.

Figure 5.3 Prediction Rejection Curves

are higher for accurate predictions than for misclassifications. An important property of this performance metric is that it is independent of classification performance, unlike AUPR.

5.3 Misclassification Detection

This section considers the use of measures of uncertainty to detect images which are misclassified by a model. These images, which will be referred to as *misclassifications*, can be considered a form of outlier and may occur for a variety of reasons. One possible reason is that they are unseen versions of seen classes, and therefore out-of-distribution relative to the training data. Alternatively, the image may be located in an in-domain region which the model does not understand well and where it has learned poor decision boundaries. In each of these situations it is desirable for the model to yield a higher uncertainty score, allowing such inputs to be detected. The current section is structured as follows: the classification performance of each model is assessed in section 5.3.1, misclassification detection is assessed as a binary classification problem in section 5.3.2 and via rejection curves in section 5.3.3.

5.3.1 Classification Performance

Firstly the classification error rate of each model trained on each data is assessed on the datasets' corresponding test set. The results are presented in table 5.4 below:

Several trends are noticeable. Firstly, MNIST, SVHN and CIFAR-10 are progressively more challenging tasks, as the error rates of all models considered consistently rise. Secondly, as the datasets become more challenging, the Prior Networks trained using the *forward* KL-divergence (PN-KL) yield an increasingly greater error rate than any of the other models,

Dataset	Single Model			Ensemble	
	DNN	PN-KL	PN-RKL	MCDP	ENS
MNIST	**0.5** ± 0.1	0.6 ± 0.1	**0.5** ± 0.1	**0.5** ± 0.1	**0.5** $\pm$ NA
SVHN	4.3 ± 0.3	5.7 ± 0.2	4.2 ± 0.2	4.3 ± 0.3	**3.3** $\pm$ NA
CIFAR-10	8.0 ± 0.4	14.7 ± 0.4	7.5 ± 0.3	8.0 ± 0.4	**6.6** $\pm$ NA

Table 5.4 Mean classification performance (% Error) $\pm 2\sigma$ across 10 random initializations. Note, performance of explicit ensemble (ENS) is not a mean, as it represents the performance of ensembling the predictions of each DNN model. PN-KL and PN-RKL trained on MNIST use the MNIST-FA out-of-distribution training data.

while Prior Networks trained using the *reverse* KL-divergence achieve the best single-model performance. These results show that on challenging tasks the *forward* KL-divergence loss yields increasingly poor classification performance. Due to this, as well as the theoretical and experimental results from the previous chapter, Prior Networks trained with *forward* KL-divergence will not be considered throughout the remained of this chapter. The final trend is that the best overall performance is always achieved using an explicit ensemble of models, which is consistent with previous work [68], while Monte-Carlo dropout ensembles do not seem to yield improvements in classification performance.

5.3.2 Assessing misclassification detection via AUPR

As described in section 5.2, misclassification detection can be seen as a threshold-based binary classification task. The performance of a model on this task can be assessed via area under an ROC curve (AUROC) or area under a Precision-Recall curve (AUPR). AUPR will be more sensitive to the performance of detecting the positive class in cases of severe class mis-balance. Given the results in table 5.4, which show that all models considered in this chapter achieve an error rate below 10%, in the following experiments misclassification detection performance is assessed only using AUPR, where misclassifications are chosen as the positive class.

The use of all uncertainty measures described in sections 3.2.1 and 3.3.1 for misclassification detection is explored. Specifically, the confidence and entropy of the predictive posterior, given in equations 3.19 and 3.18, are derived from a DNN. Both measures represent *data uncertainty*. The same measures are also derived from the predictive posterior of an explicit ensemble of DNNs, Monte-Carlo dropout ensembles and Prior Networks, where they represent *total uncertainty*. Finally, mutual information and expected pairwise KL-divergence will be derived from ensembles via equations 3.31 and 3.32 and from Prior Networks using equations 4.4 and 4.5, respectively. Furthermore, the differential entropy of a Prior Network,

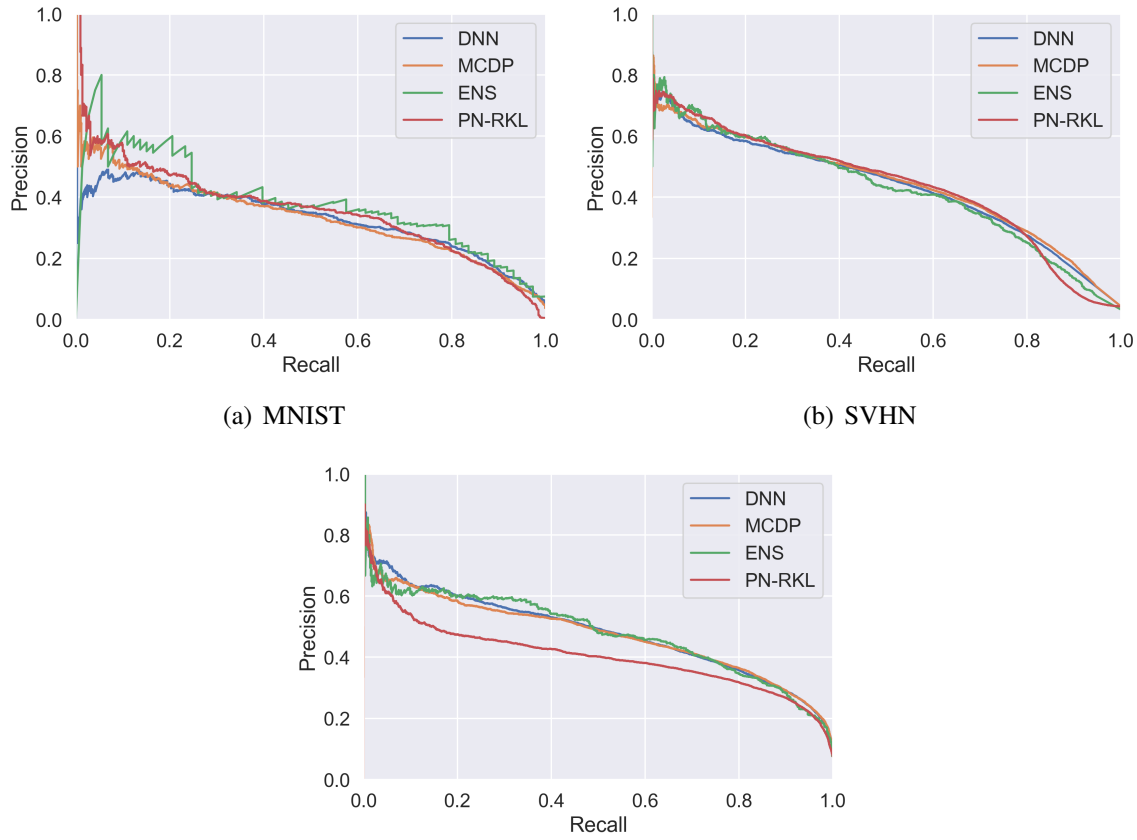
given in equation 4.6, is also considered. All of these measures of uncertainty represent *knowledge uncertainty*.

The results of the misclassification detection experiment using models trained on the MNIST, SVHN and CIFAR-10 datasets are described in table 5.1, ordered by dataset complexity. There are several notable trends in the results shown in table 5.5. Firstly, the best measure of uncertainty for misclassification detection is the confidence, while second best measure of uncertainty is always the entropy of the predictive distribution. This shows that estimates of *total uncertainty* are more useful for misclassification detection than measures of *knowledge uncertainty*, which means that it is not necessary to separate out the sources of uncertainty for this task. The reason that the confidence is a better measure of uncertainty than the entropy is because the former is directly related to the predicted class, while the latter is sensitive to changes in the entire distribution, which may be irrelevant to the given prediction. In contrast, by far the worst measures of uncertainty for this task are the expected pairwise KL-divergence and differential entropy of Prior Networks. Consequently, further analysis will only consider misclassification detection based on the confidence.

Secondly, the results show that on MNIST and SVHN all models yield similar performance. However, on CIFAR-10 Prior Networks yield the worst misclassification detection

Dataset	Model	Total Uncertainty		Knowledge Uncertainty			% Error
		Conf.	Ent.	M.I.	EPKL	D.Ent.	
MNIST	DNN	33.7 ±8.1	33.2 ±5.5	-	-	-	**0.5**
	MCDP	33.2 ±7.3	30.6 ±6.0	23.7 ±7.6	21.5 ±7.3	-	**0.5**
	ENS	**37.8** ± NA	35.5 ± NA	34.8 ± NA	34.7 ± NA	-	**0.5**
	PN-RKL	36.1 ±6.4	31.1 ±6.0	20.5 ±4.5	20.5 ±4.1	20.7 ±5.7	**0.5**
SVHN	DNN	43.4 ±3.2	43.8 ±3.5	-	-	-	4.3
	MCDP	43.3 ±3.9	**43.9** ±3.7	40.9 ±2.9	40.4 ±2.7	-	4.3
	ENS	42.7 ± NA	40.1 ± NA	37.4 ± NA	32.4 ± NA	-	**3.3**
	PN-RKL	43.8 ±4.5	43.0 ±4.4	34.9 ±3.5	34.7 ±3.5	37.1 ±3.7	4.2
CIFAR-10	DNN	48.2 ±2.7	47.0 ±3.4	-	-	-	8.0
	MCDP	**48.4** ±3.2	47.4 ±3.6	36.6 ±3.2	36.5 ±3.2	-	8.0
	ENS	48.0 ± NA	44.7 ± NA	38.6 ± NA	35.4 ± NA	-	**6.6**
	PN-RKL	40.5 ±3.0	38.5 ±2.7	35.0 ±2.3	34.8 ±2.3	36.9 ±2.6	7.5

Table 5.5 Misclassification detection results on MNIST, SVHN and CIFAR-10 test datasets in terms of mean % AUPR $\pm$ 2σ across 10 random initializations. Note, % AUPR of explicit ensemble (ENS) is not a mean, as it represents the performance of ensembling the predictions of each DNN model. MNIST Prior Network used MNIST-FA data as out-of-distribution training data.



(a) MNIST

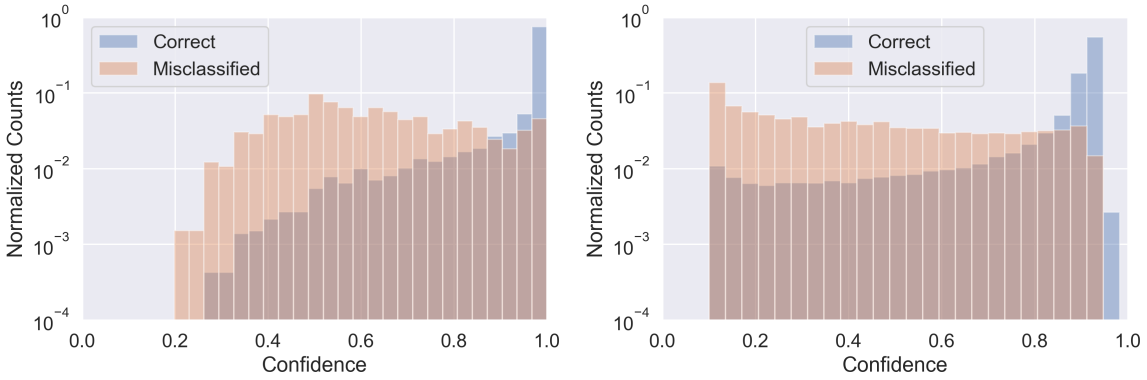
(b) SVHN

(c) CIFAR-10

Figure 5.4 Misclassification detection Precision-Recall Curves on MNIST, SVHN and CIFAR-10 test sets. Constructed using confidence as a measure of (inverse) uncertainty. The predictions of models from 10 different initializations were concatenated together as a way of combining predictions. The exception to this is the explicit ensemble (ENS), where only a single set of predictions is available.

performance by a large margin. This can be further analyzed by considering the PR-curves depicted in figure 5.4. The PR curves derived from all models on the MNIST and SVHN are similar, though the PR curves on MNIST are noisy as there are so few misclassifications. However, the PR curves for models trained on CIFAR-10, depicted in figure 5.4c, show that Prior Networks yield consistently lower precision, especially as the uncertainty threshold is increased.

Let's consider the histograms of the confidence scores for correct and misclassified inputs of an explicit ensemble and a Prior Network trained on CIFAR-10, shown in figure 5.5. By comparing figures 5.5a and 5.5b it is clear that Prior Networks tend to make more classification with a low confidence than ensembles. These result suggest that due to overall similarity between the CIFAR-10 and CIFAR-100 data, the out-of-distribution loss decreases



(a) Explicit Ensemble (b) Prior Network

Figure 5.5 Histograms of confidence scores of explicit ensemble (ENS) and Prior Network (PN-RKL) on CIFAR-10 test set. The predictions of models from 10 PN-RKL models trained form different initializations were concatenated together as a way of combining predictions. Only a single set of predictions is available from ENS.

the confidence the model assigns to inputs which are close to the CIFAR-10 region. However, considering that the overall error rate goes down, it is possible there are two separate effects occurring. Firstly, the model makes more low-confidence predictions overall due to the effect of CIFAR-100 data. Secondly, low-confidence misclassifications have now become low-confidence correct predictions. As a result, there is a greater number of misclassification and correct predictions which have a low confidence, which will bring down the precision, decreasing overall AUPR, as shown in figure 5.4c. In contrast, models trained on MNIST and SVHN used out-of-distribution training data which was quite different from the in-domain data and which therefore had little effect on in-domain performance.

These results show that in general, Prior Networks do not provide a significant advantage to misclassification detection over standard models like DNN and ensemble of DNNs. However, the out-of-distribution data can act as a regularizer and improves overall classification performance. Finally, if there are similarities between the out-of-distribution training data and the in-domain data, then it can actually decrease the precision with which misclassifications can be detected and shift the overall level of confidence down.

5.3.3 Assessing misclassification detection via rejection curves

As discussed in section 5.2, an alternative way to assess misclassification detection performance is via rejection curves, which represent a realistic downstream application of uncertainty estimates. Here, the predictions a model makes are replaced by the predictions of an ‘oracle’, for example a human, in order of decreasing uncertainty. The rejection curves

Dataset	Single Model		Ensemble	
	DNN	PN-RKL	MCDP	ENS
MNIST	98.2 ± 0.3	95.8 ± 1.5	97.9 ± 0.5	**98.4** $\pm$ NA
SVHN	80.7 ± 2.1	77.8 ± 2.0	81.6 ± 2.1	**84.4** $\pm$ NA
CIFAR-10	84.4 ± 1.1	82.0 ± 1.4	84.4 ± 1.1	**86.9** $\pm$ NA

Table 5.6 Mean rejection ratios $\pm 2\sigma$ across 10 random initializations on MNIST, SVHN and CIFAR-10 test sets. Constructed using confidence as a measures of (inverse) uncertainty. Performance is evaluated on the test sets of MNIST, SVHN and CIFAR-10.

for all models trained on the MNIST, SVHN and CIFAR-10 datasets are shown in figure 5.6. Predictions are rejected in order of decreasing confidence. The rejection performance can be summarized by the rejection area ratio, introduced in section 5.2. These ratios are provided in table 5.6. The results show that the best model at rejection is an explicit ensemble, while Prior Networks consistently yield the worst rejection performance. This essentially reflects the same trends as the analysis based on precision-recall curves above. Analyzing precision-recall curves is more informative than rejection curves. However, rejection curves provide more information about performance on a downstream application. Furthermore, it may not be possible to use precision-recall curves to analyze the quality of uncertainty estimates on certain tasks, like regression. In these situations it is possible to use rejection curves and rejection area ratios instead.

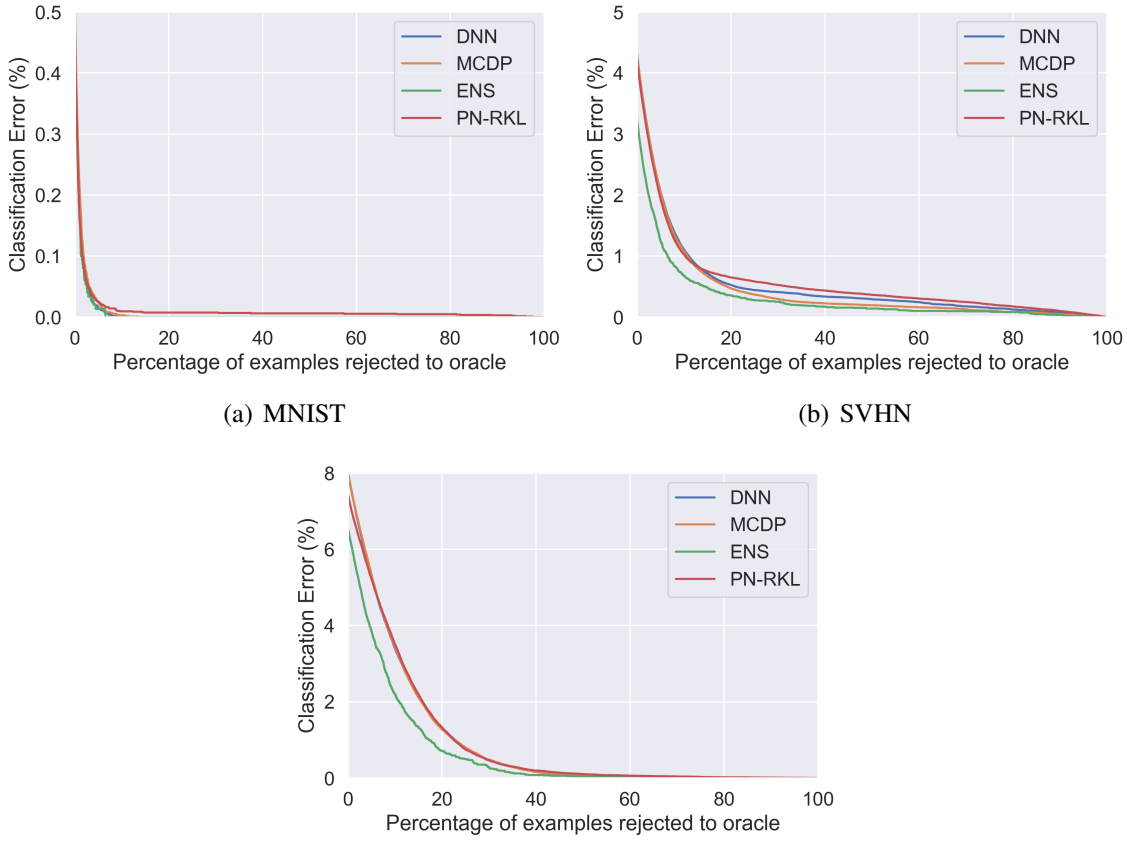
5.4 Out-of-Distribution sample Detection

The previous section investigates using measures of uncertainty to detect misclassifications, which were considered to be a form of outlier. Another form of outlier are out-of-distribution or out-of-domain inputs, which come from a different distribution than the one which the model was trained on. As discussed in section 3.1.1, in the context of classification out-of-distribution inputs can either be unseen variations for known classes or examples of completely unknown classes. Given an out-of-distribution input the model may yield nonsensical predictions and have generally undefined behaviour. Consequently, it is desired that a model understands that it has been exposed to out-of-distribution inputs and indicates this by producing a high estimate of uncertainty in predictions. This allows the model to able to detect out-of-distribution inputs and appropriate actions to be taken. Thus, the detection of out-of-distribution inputs based on measures of uncertainty is investigated in this section.

In the experiments described here in-domain test images are paired with an equal number[1] of out-of-distribution images taken from a *different* dataset. Given this combined dataset, the goal is to detect which images are out-of-distribution based on uncertainty estimates. The performance on this binary classification task can be assessed both via area under the ROC curve or area under Precision-Recall curve. However, as the experiment is designed such that there is equal number of out-of-distribution and in-domain images, area under an ROC curve (AUROC) is used as a metric of performance. The measures of uncertainty considered for misclassification detection in the previous sections are also explored for out-of-distribution input detection.

This section is structured as follow: out-of-distribution detection using models trained on MNIST is investigated in section 5.4.1. Out-of-distribution detection using models trained

[1] With one exception.



(a) MNIST

(b) SVHN

(c) CIFAR

Figure 5.6 Mean rejection curves across 10 random initializations on MNIST, SVHN and CIFAR-10. Constructed using confidence as a measures of (inverse) uncertainty. Note, only a single set of predictions is obtained from explicit ensemble ENS.

on CIFAR-10 is investigated in section 5.4.2. Experiments on models trained on SVHN are presented in appendix C.

5.4.1 MNIST out-of-distribution input detection

In this section the estimates of uncertainty provided by models trained on the MNIST dataset are evaluated on the task of discriminating between the in-domain MNIST dataset and the out-of-distribution Semeion and Omniglot datasets. These out of-distribution datasets are described in tables 5.1 and 5.2 and depicted in figure 5.1. Semeion is a dataset of black-and-white handwritten digits whose primary difference from MNIST is that there is no padding between the edge of the image and the digit, while Omniglot is a dataset of black-and-white handwritten characters from a range of different alphabets. Note, as Semeion is a very small dataset it was not possible to get a balanced set of MNIST and Semeion images. Thus, AUPR is used instead of AUROC as a metric of performance when discriminating between MNIST and Semeion. The results of these experiments are presented in table 5.7.

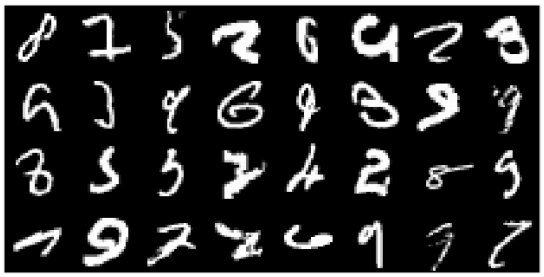
There are several notable trends. Firstly, measures of *knowledge uncertainty* are usually better at detecting OOD images than measures of *total uncertainty*, though the difference is small. This makes sense, as MNIST is a low *data uncertainty* dataset, indicated by the low classification error rate, and it was shown in section 4.2.3 that all measures of uncertainty yield similar behaviour on a low *data uncertainty* dataset. Curiously, it seems that expected pairwise KL-divergence of an ensemble yields marginally better performance than mutual information. At the same time, the overall worst measure of uncertainty for

OOD Data	Model	Total Uncertainty		Knowledge Uncertainty		
		Conf.	Ent.	M.I.	EPKL	D.Ent.
Semeion*	DNN	77.3 $\pm$3.7	78.2 $\pm$3.7	-	-	-
	MCDP	83.7 $\pm$1.9	84.8 $\pm$2.1	88.0 $\pm$1.3	88.6 $\pm$1.2	-
	ENS	83.6 $\pm$ NA	84.5 $\pm$ NA	90.2 $\pm$ NA	90.7 $\pm$ NA	-
	PN-RKL	96.5 $\pm$0.6	97.6 $\pm$0.6	99.2 $\pm$0.4	**99.3** $\pm$0.4	95.4 $\pm$3.9
Omni-EVAL	DNN	97.8 $\pm$0.3	97.9 $\pm$0.3	-	-	-
	MCDP	97.9 $\pm$0.2	98.0 $\pm$0.2	97.8 $\pm$0.2	97.7 $\pm$0.2	-
	ENS	98.2 $\pm$ NA	98.3 $\pm$ NA	98.5 $\pm$ NA	**98.6** $\pm$ NA	-
	PN-RKL	97.7 $\pm$0.5	97.7 $\pm$0.5	94.1 $\pm$1.3	94.1 $\pm$1.3	92.4 $\pm$2.0

Table 5.7 MNIST out-of-domain detection results. Results against Semeion are quoted in terms of mean % AUPR $\pm 2\sigma$ across 10 random initializations, as there are few Semeion images. Results against Omniglot eval dataset are quoted in terms of mean % AUROC $\pm 2\sigma$, as there are roughly equal amounts of MNIST (test+valid) and Omniglot images.

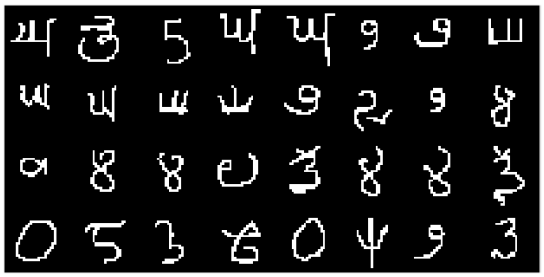
OOD detection is the confidence in the prediction. This is the exact opposite behaviour to misclassification detection, where confidence yielded the best performance and measures of *knowledge uncertainty* the worst performance. Secondly, Prior Networks yield the highest performance by a large margin on the Semeion dataset in comparison to the baselines. However, on the Omniglot dataset all model yield similar, high levels of performance, though Prior Networks do worst. Finally, explicit ensembles of DNNs are consistently the strongest baseline and outperform ensembles generated via Monte-Carlo dropout, which is consistent with previous work [68].

These results can be further analyzed by visualizing the *highest* uncertainty in-domain (MNIST) images and the *lowest* uncertainty out-of-distribution (Semeion/Omniglot) images. Figure 5.7 shows that Semeion digits are much more 'chunky' than MNIST images, while the lowest uncertainty Omniglot images has a style much closer to that of MNIST. Furthermore, the lowest uncertainty Omniglot images are characters which are similar to the digits 0-9. Clearly, a model trained on MNIST would find it difficult to consider characters which look like digits as out-of-distribution. This suggests that the out-of-distribution training data, generated via Factor analysis, is too crude to capture the difference between these Omniglot images and MNIST.



(a) High uncertainty MNIST

(b) Low uncertainty Semeion



(c) Low uncertainty Omniglot

Figure 5.7 Highest mutual information in-domain (MNIST test set) images and lowest mutual information out-of-domain Semeion and Omniglot images.

Thus, it is interesting to investigate the performance of a Prior Network trained on the Omniglot 'background' datasets BGS1, BGS2 and BG, which contains different, non-overlapping sets of characters than the Omniglot evaluation dataset. There are several trends in the results presented in table 5.8. Firstly, OOD training data generated via Factor analysis drives the Prior Networks to better discriminate between MNIST and Semeion images, although Omniglot data does not do much worse. Furthermore, increasing the size of the Omniglot OOD training data from BGS1 to BGS1+2 to finally the entire background set BG improves this performance. Secondly, training a Prior Network using even the smaller Omniglot background dataset BGS1 allows it to perfectly discriminate between MNIST and Omniglot images. It is important to stress that the characters and alphabets in the Omniglot background and evaluation dataset are different and non-overlapping. These results show that given the ability of Prior Networks to discriminate between in-domain and out-of-distribution images strongly depends on the out-of-distribution training data. Appropriate choice of this data can allow Prior Networks to achieve very high performance.

Out-of-Domain Data		Total Uncertainty		Knowledge Uncertainty		
Test	Training	Conf	Ent.	M.I.	EPKL	D.Ent.
Semeion*	MNIST-FA	96.5 $\pm$0.6	97.6 $\pm$0.6	99.2 $\pm$0.4	**99.3** $\pm$0.4	95.4 $\pm$3.9
	Omni-BG1	95.1 $\pm$1.5	96.2 $\pm$1.3	98.0 $\pm$1.0	98.2 $\pm$0.9	92.9 $\pm$3.5
	Omni-BG1+2	95.2 $\pm$1.2	96.3 $\pm$1.2	98.1 $\pm$0.9	98.3 $\pm$0.8	92.7 $\pm$3.4
	Omni-BG	95.4 $\pm$1.3	96.5 $\pm$1.2	98.1 $\pm$0.8	98.3 $\pm$0.7	93.1 $\pm$3.3
Omni-EVAL	MNIST-FA	97.7 $\pm$0.5	97.7 $\pm$0.5	94.1 $\pm$1.3	94.1 $\pm$1.3	92.4 $\pm$2.0
	Omni-BGS1					
	Omni-BGS1+2	**100.0** $\pm$0.0	**100.0** $\pm$0.0	**100.0** $\pm$0.0	**100.0** $\pm$0.0	**100.0** $\pm$0.0
	Omni-BG					

Table 5.8 Investigation of effect of out-of-distribution training data on out-of-domain detection performance of Prior Networks. Results against SEMEION are quoted in terms of mean % AUPR $\pm 2\sigma$ across 10 random initializations, as there are few SEMEION images. Results against Omniglot eval dataset are quoted in terms of mean % AUROC $\pm 2\sigma$, as there are roughly equal amounts of MNIST (test+valid) and Omniglot images.

5.4.2 CIFAR-10 out-of-distribution input detection

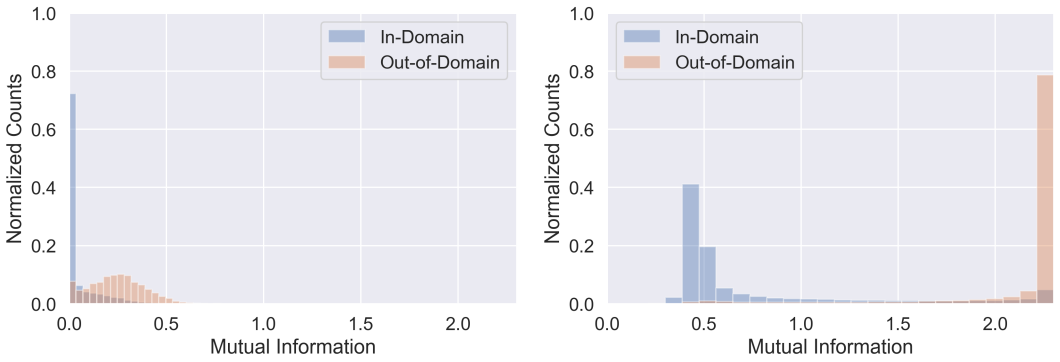
In the previous section it was shown that Prior Networks trained on MNIST can successfully detect out-of-distribution images. However, MNIST is a very easy task, and it is necessary to investigate whether the Prior Networks scale to more complex tasks, such as CIFAR-10. The experiments described in this section evaluate the ability of models trained on CIFAR-10 to

discriminate between in-domain images from the CIFAR-10 test set and out-of-distribution images from the test sets of SVHN, LSUN and TinyImageNet. This is a more challenging task, as there are far more possible factor of variation in 32×32 color images of real objects than in black-and-white images of handwritten characters. The result of these experiments are presented in table 5.9.

OOD Data	Model	Total Uncertainty		Knowledge Uncertainty		
		Conf.	Ent.	M.I.	EPKL	D.Ent.
SVHN	DNN	90.3 $\pm$1.9	91.0 $\pm$2.1	-	-	-
	MCDP	90.2 $\pm$1.9	91.0 $\pm$2.1	88.9 $\pm$1.8	88.9 $\pm$1.8	-
	ENS	92.0 $\pm$ NA	92.9 $\pm$ NA	89.5 $\pm$ NA	89.0 $\pm$ NA	-
	PN-RKL	98.1 $\pm$1.2	98.2 $\pm$1.1	98.3 $\pm$0.9	**98.3** $\pm$0.9	98.2 $\pm$1.1
LSUN	DNN	89.8 $\pm$0.9	91.4 $\pm$0.9	-	-	-
	MCDP	89.7 $\pm$0.9	91.2 $\pm$0.9	89.9 $\pm$1.2	89.9 $\pm$1.2	-
	ENS	92.5 $\pm$ NA	94.3 $\pm$ NA	93.2 $\pm$ NA	92.5 $\pm$ NA	-
	PN-RKL	95.6 $\pm$0.9	95.7 $\pm$0.9	95.8 $\pm$0.8	**95.8** $\pm$0.8	**95.8** $\pm$0.8
TIM	DNN	87.6 $\pm$2.2	88.8 $\pm$2.3	-	-	-
	MCDP	87.4 $\pm$2.2	88.6 $\pm$2.3	87.2 $\pm$1.5	87.2 $\pm$1.5	-
	ENS	90.0 $\pm$ NA	91.5 $\pm$ NA	90.3 $\pm$ NA	89.7 $\pm$ NA	-
	PN-RKL	95.6 $\pm$0.7	95.7 $\pm$0.7	95.8 $\pm$0.7	**95.8** $\pm$0.7	**95.8** $\pm$0.7

Table 5.9 CIFAR-10 out-of-domain detection results in terms of mean % AUROC $\pm 2\sigma$ across 10 random initializations (except ENS). CIFAR-10 test set is used as in-domain data and the test sets of SVHN (10,000 image subset), LSUN and TIM as out-of-domain data.

There are several trends in the results. Firstly, the results show that Prior Networks consistently outperform the baselines approaches on all tasks and using all measures of uncertainty. The best baseline approach, as in previous experiments, is an explicit ensemble of DNNs. At the same time, ensembles generated via Monte-Carlo dropout yield the same level of performance as standard DNNs trained with maximum likelihood. The results show that, based on the performance of baseline approaches, the most challenging out-of-distribution dataset is TinyImageNet. This is expected, as it contains images of a wide variety of real object categories and is the most similar to CIFAR-10. As before, measures of *knowledge uncertainty* derived from a Prior Network, specifically the expected pairwise KL-divergence and differential entropy, yield the best results. However, the advantage over the other measures is small as CIFAR-10, like MNIST, is also a low *data uncertainty* dataset. Curiously, the best measure of uncertainty derived from ensembles is the entropy of the predictive posterior, a measure of *total uncertainty*. It is not clear why this is the case, but



(a) ENS (b) PN-RKL

Figure 5.8 Histogram of mutual information for in-domain (CIFAR-10 test set) and out-of-domain (TinyImageNet test set) images derived from explicit ensemble (ENS) and Prior Network (PN-RKL). Predictions of 10 PN-RKL models trained from different random initializations are concatenated together.

supports the assertion that it is difficult to control the behaviour of ensembles such that they are diverse out-of-domain.

It is interesting to analyze the difference in the behaviour of ensembles and Prior Networks in greater detail. Consider figure 5.8, which depicts the histograms of estimates of mutual information derived from an ensemble and Prior Networks for CIFAR-10 and TinyImageNet images. Figure 5.8 shows that the ensemble yields a much tighter distribution of mutual information and is less 'diverse' than a Prior Network. There is also a greater region of overlap between in-domain images and out-of-domain images. At the same time, the Prior Network clearly separates out the in-domain and out-of-distribution images. However, there are a small number of out-of-distributions images for which the Prior Network yields low uncertainty and a set of in-domain images for which the Prior Network yields maximum uncertainty.

These images are depicted in figure 5.9. Figure 5.9a shows that the high-uncertainty in-domain images are 'odd' images with obstructions, strange backgrounds, etc, making it difficult to actually determine what they represent. As discussed in section 5.3.2, Prior Networks make more low-confidence predictions than ensembles, which is likely a result of out-of-distribution data, in this case CIFAR-100, being close to the in-domain data in certain regions. Improving the performance of the classifier and better choice of out-of-distribution training data could possibly decrease the number of high uncertainty in-domain images which are not misclassified. At the same time, figure 5.9b shows that the low uncertainty out-of-distribution images are images of dogs, cats and cars, which are classes present in the

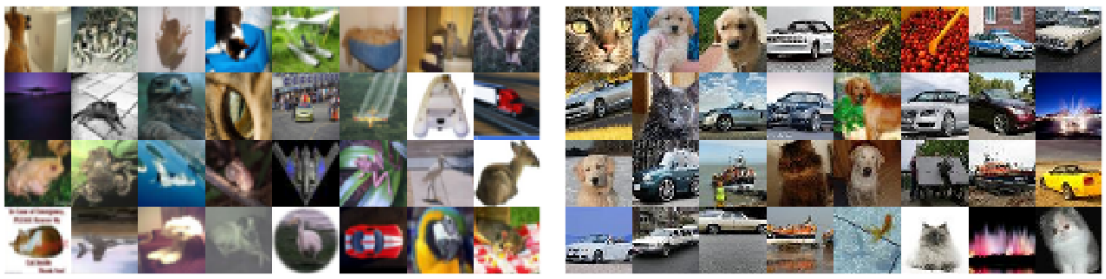
CIFAR-10 dataset. Thus, there is a partial overlap between TinyImageNet and CIFAR-10 classes.

5.5 Chapter Summary

In this chapter the construction of Prior Networks on the MNIST, SVHN and CIFAR-10 image classification datasets was investigated. Measures of uncertainty derived from Prior Networks trained on these datasets were evaluated on the tasks of misclassification detection and out-of-distribution sample detection. Measures of uncertainty derived from standard DNNs, a Monte-Carlo dropout ensemble and an explicit ensemble of models were used as baselines.

In section 5.3 it was shown that Prior Networks do not provide a performance gain over baseline approaches on the tasks on misclassification detection. Furthermore, misclassification detection performance of Prior Networks can be degraded if the out-of-distribution training data is too close to the in-domain region. At the same time, the out-of-distribution training data can also act as a regularizer and improve classification performance of Prior Networks. The results in section show that an explicit ensemble of models yields both the lowest classification error rates and the best misclassification detection performance. It was determined that the confidence in the predictions (the probability of the mode of the predictive distribution) is consistently the best measure of uncertainty for misclassification detection.

Results in section 5.4 show that Prior Networks are able to outperform baseline approaches on the task of out-of-distribution input detection on both the MNIST and CIFAR-10 dataset. Results on the SVHN dataset, available in appendix C, also show that a Prior Network is capable outperforming baseline approaches by a large margin. The best baseline



(a) High Uncertainty CIFAR-10 images (b) Low Uncertainty TinyImagenet images

Figure 5.9 Highest mutual information in-domain (CIFAR-10 test set) images and lowest mutual information out-of-domain TinyImageNet images.

approach to uncertainty estimation throughout all experiments was an explicit ensemble of DNNs. The results in this section also show that measures of *knowledge uncertainty*, such as mutual information, expected pairwise KL-divergence and differential entropy, yield better out-of-distribution detection results, which illustrates the benefit of being able to decompose *total uncertainty* into *data* and *knowledge uncertainty*.

The superior out-of-distribution detection performance of Prior Networks supports the assertion that it is easier to control the out-of-distribution behaviour of a distribution over distributions via choice of training data, rather than by appropriate choice of prior and approximate inference method. However, the choice of out-of-distribution training data is non-trivial and needs to be further explored in detail. Furthermore, the low classification error rates and superior misclassification detection performance of explicit ensembles of models indicates that appropriate choice of 'out-of-distribution' data which represents other forms of dataset shift also needs to be explored. As a Prior Network represents a distribution over distributions and is theoretically capable of emulating an ensemble, a possible avenue of future work would be to distill an ensemble into a Prior Network, combining the advantages of both. Additionally, future work should investigate the application of Prior Networks to more complex datasets, such as ImageNet [27], which have more than 10 classes, and to structured data, such as language and speech. Active learning applications of Prior Networks should also be investigated in order to further evaluate the practical benefit of separating out the sources of uncertainty. Finally, Prior Networks should also be evaluated on a range of regressions tasks.

This chapter concludes the first part of this thesis, which focused on uncertainty estimation. The next three chapters will consider the application of uncertainty estimation to the areas of spoken language proficiency assessment.

Chapter 6

Spoken Language Proficiency Assessment

The first part of this thesis explored the area of predictive uncertainty estimation, discussed single model and ensemble based approaches in chapter 3, proposed a new class of models called Prior Networks in chapter 4 and compared them to previous approaches in chapter 5. This chapter begins the second part of this thesis, which investigates the application of deep learning and uncertainty estimation to the area of automatic *spoken* language assessment, specifically automatic grading of spoken proficiency examinations and automatic assessment of relevance of spoken responses to open-ended exam prompts.

The increasing demand for language learning and for practice tests available at any time make the development of automatic assessment systems an attractive proposition [110]. The goal of an automatic grader is to assess language competence of an examination candidate with an accuracy matching that of a human grader, but faster, with greater consistency and at a fraction of the cost. As mentioned in the introduction of this thesis, automatic assessment, especially high-stakes automatic assessment, requires that models provide measures of uncertainty in their predictions in order to avoid making mistakes which can adversely affect the course of peoples' lives, as was the case with the Irish veterinarian who failed an automatically assessed spoken English exam in Australia due to her accent [1]. This motivates the application of the approaches to uncertainty estimation, discussed in the first part of this thesis, to the tasks of automatic grading and automatic relevance assessment, which is considered the second part of this thesis.

The current chapter introduces the area of automatic assessment of non-native spoken language proficiency and discusses the attributes and challenges associated with this task. This chapter is structured as follows: section 6.1 describes the overall task of spoken language assessment, grade levels and attributes of the BULATS and LinguaSkill exams, provided by

Cambridge English Language Assessment. Section 6.2 describes the automatic assessment pipeline used in this work as well the tasks of automatic grading and automatic prompt-response relevance assessment.

6.1 Spoken Language Proficiency

There is a high demand around the world for learning English as a second language and, therefore, a need to assess the proficiency level of learners both during their studies and for formal qualifications. Language proficiency is assessed using standardized, universally accepted, examinations, such as International English Language Testing System (IELTS) and Test of English as a Foreign Language (TOEFL), which aim to provide a holistic assessment of a candidate's language proficiency. These tests often include listening, speaking, reading and writing sections that are graded by well-trained human examiners who assign a score based on a set of guidelines.

A key part of learning a language is learning how to speak fluently and with confidence. The level reached can be assessed through *spoken language* proficiency exams, where candidates are prompted to respond to a series of open-ended questions, such as "describe a difficult situation at work, why was it difficult and how did you resolve it?". Traditionally, human examiners assess the candidate's spontaneous speech replies in terms of pronunciation, hesitations/extent, use of grammar and vocabulary, and coherency of discourse. Spoken language proficiency is typically assessed according to the *Common European Framework of Reference for Languages* (CEFR), which is designed to provide a basis for the elaboration of language syllabi, the design of teaching and learning materials, and the assessment of foreign language proficiency [96, 124]. The CEFR framework separates learners into three broad categories which can be divided into six proficiency levels detailed in table 6.1:

Level Group	Group Name	Level	Level Description
A	Basic User	A1	Breakthrough or beginner
		A2	Waystage or elementary
B	Independent User	B1	Threshold or intermediate
		B2	Vantage or upper intermediate
C	Proficient User	C1	Effective Operational Proficiency
		C2	Mastery or Advanced Proficiency

Table 6.1 CEFR Foreign Language Proficiency Levels

The CEFR framework provides qualitative measures for reception, interaction and production to distinguish between the different levels. A C1 level learner can understand extended

speech even when it is not clearly structured, and when relationships are only implied and not signalled explicitly. A B1 learner can only understand the main points of clear standard speech on familiar matters regularly encountered in work, school or leisure, for example. Similarly for speech production, an A1 user can use simple phrases and sentences to describe their surroundings whereas a C2 learner is able to present a clear, smoothly-flowing description or an argument with an effective logical structure.

This work considers the assessment of spoken language proficiency of candidates taking the Business Language Testing Service (BULATS) and LinguaSkill exams provided by Cambridge English Language Assessment. The BULATS exam is a set of workplace language assessment, training and bench-marking tools that is used internationally: for business and industry recruitment; to identify and deliver training; for admission to study business-related courses; and for assessing the effectiveness of language courses [15]. Linguaskill is an online, multi-level test which is designed to offer a complete picture of a candidates' English abilities, with fast and accurate testing of all four language skills: reading, listening, writing, and speaking. The primary difference between BULATS and LinguaSkill is that BULATS is focused on prompts related to business, industry and commerce while LinguaSkill covers a broad range of subjects including, but not limited to: family and friends, travel and holidays, places and sights, studying and working, shopping, sport and music. The overall structure of both the BULATS and LinguaSkill exams is similar - both exams have five sections:

1. the candidate answers eight questions about themselves (e.g. what is your name, where do you come from?);
2. candidates read aloud six (or eight) short texts;
3. the candidate is given a particular topic to talk about (e.g. the perfect office, holiday, etc...);
4. candidates must describe one or more graphics (such as a diagram or information sheet) related to a topic (stock prices, demographics, etc..);
5. candidates are asked to respond to five open-ended questions related to a single context prompt. For example a set of five questions about organizing a stall at a trade fair.

The first section (A) contains eight questions about the candidate (e.g. "How do you use English in your job?"). The second section (B) is a read-aloud section in which the candidates are asked to read eight sentences. The last three sections (C, D and E) have longer utterances of spontaneous speech elicited by prompts. In section C the candidates are asked to talk for one minute about a prompted topic. In section D, the candidate has one minute to describe a

situation illustrated in graphs or charts, such as pie or bar charts. The prompt for section E asks the candidate to imagine they are in a specific conversation and to respond to questions they may be asked in that situation (e.g. advice about planning a conference). Each section of the BULATS or LinguaSkill exam is manually scored between 0 and 6 by a trained human examiner; the overall score is therefore between 0 and 30. These can be mapped into CEFR (Common European Framework of Reference) ability levels, described in table 6.1, as detailed in table 6.2.

BULATS score range	CEFR level
29-30	C2
25-28.5	C1
20-24.5	B2
15-19.5	B1
10-14.5	A2
5-9.5	A1
0-4.5	pre-A1

Table 6.2 Equivalence between BULATS/LinguaSkill scores and CEFR levels.

6.2 Automatic Assessment

To meet demand from English learners, the introduction of automatic graders for spoken language assessment would be beneficial, especially for practice situations. As previously stated, the goal of an automatic grader is to assess language competence and provide scores reflecting the quality of the responses given by the candidates in a manner emulating the accuracy that could be achieved by a human grader. Compared to human graders, automated graders potentially perform more consistently and offer faster feedback times at a fraction of the marginal cost. In comparison, the process of hiring and training new a human grader is costly and only offers a small increase in throughput.

The pipeline for a typical automatic spoken language proficiency assessment system [25, 128, 33, 34, 129, 137, 119, 85] is shown in figure 6.1. Similar to a number of other systems, including [25, 33, 137, 52], the graders considered in this thesis use a series of features based on the speaker's audio and fluency. Audio features are extracted directly from the audio signal. However, audio features alone do not contain sufficient information to represent the candidates' English proficiency. Automatic assessment systems additionally use an automatic speech recognition (ASR) system to obtain automatic transcriptions, or hypotheses, of the candidates' responses to exam questions. Features for speaker fluency,

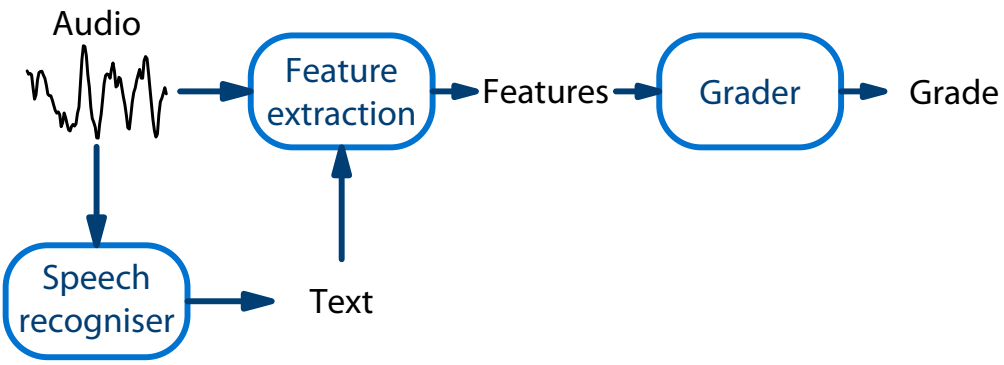


Figure 6.1 CUED Automatic Spoken Language Assessment Pipeline

such as the speaking rate and mean duration of words and silences are derived from the ASR hypotheses, time aligned to the audio. A number of the individual features show high correlation with the scores, the remainder have been found to contribute to grader performance when used in combination with other features [126]. These statistics are derived from all responses given by a candidate. Compared to the audio features that are used in [119], in this work 5 new features were added relating to disfluencies, recording duration and vowel frequency, leading to a 33-dimensional feature set described in table 6.3. An automatic grader is trained on these features with grade-targets provided by human examiners. More diverse features are investigated in [126], however, in this thesis only the features in table 6.3 will be used.

An example of a state-of-the-art grader trained on data derived from candidates' responses to the BULATs exam is a Gaussian Process grader [119]. Automatic grading can be treated both as a classification, where the model predicts a CEFR grade-level for each candidate, and as a regression task, where the model predicts a real value 'score' which can then be mapped to a CEFR grade. However, due to operational requirements of Cambridge English Language Assessment, in this thesis automatic grading will only be investigated as a regression task.

Robust automatic assessment is a challenging task for a number of reasons. Human examiners assign grades based on a holistic assessment of a candidate's spoken responses in terms of pronunciation, hesitations and their extent, use of grammar and vocabulary, coherency of discourse, response content and understanding of the question. Unlike automatic assessment of *written* proficiency [132], where an automatic system has access to the same information as a human grader - the candidate's written response, automatic assessment systems for speech do not have direct access to candidate's spoken response. Instead, they operate on a set of shallow proxy features derived from the candidate's spoken response, such as the ones detailed in table 6.3. While these features *correlate* well with spoken language proficiency, they clearly do not capture the *semantic content* of a spoken response, and

No.	Item	Statistics
	Audio Features	
1	f_0	mean
2		normalised minimum value
3		normalised maximum value
4		normalised range
5		normalised mean absolute deviation
6	Energy	mean
7		normalised minimum value
8		normalised maximum value
9		normalised standard deviation
10		normalised mean absolute deviation
	Text (Fluency) Features	
11	Long Silence Duration	mean
12		standard deviation
13		median
14		mean absolute deviation
15	disfluencies	number of partial words
16		number all
17		fraction all
18	hesitations	number
19		fraction
20	Long Silences	number
21	Phone/Grapheme Duration	mean
22		standard deviation
23		median
24		mean absolute deviation
25	Silence Duration	mean
26		standard deviation
27		median
28		mean absolute deviation
29	Words	number
30		frequency
31		mean duration
32	Time Used	fraction
33	Syllables	frequency

Table 6.3 CUED Grader Features

therefore cannot holistically assess the candidate's understanding of the question, sentence construction, and discourse coherence.

This raises several concerns about the validity of automatic assessment considering that three out of five sections on the BULATS and LinguaSkill exams, detailed in the previous section, consist of open-ended questions which elicit unstructured, spontaneous speech. Crucially, the inability to assess the *relevance* of a candidate's response to a question potentially opens the system to spoofing by malicious candidates who wish to achieve a high score without actually being proficient - they may memorize some text and deliver it fluently and with confidence rather than actually responding to the prompts of the exam. This creates a need to assess the *relevance* of a response to exam prompts.

Another challenge of automatic assessment is the difficulty of robustly deriving these audio and fluency features from a candidate's spoken response. The speech to be scored should contain spontaneous sections instead of simply be readings of a known text, in order to be able to assess understanding and sentence coherency. This introduces difficulties to the ASR system because spontaneous speech normally contains disfluencies such as false starts, hesitations and partial words, increasing the ASR error rate. Furthermore, non-native spontaneous speech is accented and contains grammatical and sentence constructions errors whose the degree and nature are strongly impacted by the first language (L1) and proficiency level of the candidates. Finally, the levels of background noise and volume levels of the audio recordings are likely to vary by a large amount due to variation in recording conditions in across exam centers in different countries. In short, each stage of the automatic assessment pipeline depicted in figure 6.1, from recording the audio to the derivation of features, adds a certain amount of noise and distortion which is affected by the nature of the speaker and recording conditions. As a result, the information contained in the features strongly depends on multiple variables which are difficult to know at test time. Furthermore, it is in practice impossible observe all of these variations in a finite training dataset. Thus, the predicted grade's validity will decrease the more the candidate is mismatched to the data used to train the system. This is a clear example of *dataset shift* [102], a term use to describe mismatches between training and test dataset, which is difficult to account for. Thus, if these automatic assessment systems are to be safely deployed to high-stakes tests, it is necessary to have estimates of the system's uncertainty in its predictions. Knowing when the system is uncertain allows the detection of "outlier" candidates who need to be examined by, for example, human graders. This can allow automatic assessment system to avoid situations like the one where an Irish veterinarian failed an automatic spoken language proficiency exam in Australia due to accent mismatch [1].

6.3 Chapter Summary

The current chapter transitioned from the first part of this thesis, which discussed uncertainty estimation for deep learning, to the second part, which considers the application of deep learning and uncertainty estimation to the area of automatic assessment of spoken language proficiency. Section 6.1 discussed assessment of spoken language proficiency and described the BULATS and LinguaSkill exams, provided by Cambridge English Language Assessment. Datasets derived from these two exams will be used in the next two chapters. Section 6.2 described the standard pipeline for automatic assessment and the typical features used, and discussed the challenges of automatic assessment. Specifically, the limited and shallow nature of the the features, which does not allow the assessment of the relevance of spoken responses to exam questions, and the difficulty of their robust estimation, which leads to significant dataset shift, are discussed. This motivates the investigation of deep-learning based automatic graders which provide estimates of uncertainty and models for the assessment of the relevance of spoken responses to exam prompts based on ASR transcriptions in chapters 7 and 8, respectively.

Chapter 7

Deep Learning for Automatic Grading

The previous chapter discussed the area of spoken language assessment and the associated challenges, motivating the tasks of automatic grading and automatic assessment of relevance of spoken responses to open-ended prompts. The current chapter investigates the construction of models for automatic grading of spoken language proficiency examinations using neural networks and compares them to baseline models, such as Gaussian Processes [119]. The models are evaluated on how well their grades match a set of expert human grades on a test set.

One of the challenges of automatic assessment is that human examiners assign grades based on a holistic assessment of the candidates' spoken responses, while automatic graders must predict the same grade based a set of shallow proxy features derived from the response. As discussed in the previous chapter, the recording conditions, as well as the accent, first language and proficiency level or the candidates can have an immense impact on the feature extraction process, which affects how well an automatic grader performs. This makes it highly desirable for automatic graders to yield estimates of uncertainty in their predictions, especially if they are used for high-stakes examinations. Techniques for deriving uncertainty estimates discussed in chapter 3 will be applied to the models developed in this chapter. Specifically, uncertainty estimates provided by standard Density Networks, ensembles of Density Networks and Density Networks trained in a multi-task fashion to yield high uncertainty for out-of-domain inputs are evaluated and compared to measures of uncertainty provided by Gaussian Process graders. These uncertainty estimates will be evaluated in two ways. Firstly, we consider the scenario where an automatic grader passes over an exam candidate to a human examiner based on its estimates of uncertainty. The goal is to pass over the smallest number of candidates on which the automatic grader makes the biggest errors. Secondly, uncertainty estimates will be used to derive *predictive intervals* on the models' predictions. The quality of the predictive intervals will be assessed via calibration curves.

This chapter is be structured as follows: section 7.1 discusses the automatic assessment pipeline used in this chapter and provides a description of Gaussian Process and deep-learning based models for automatic grading; in section 7.2 these approaches are evaluated on data from the BULATS spoken language proficiency exams in terms of their grading performance and the quality of the uncertainty estimates they provide.

7.1 Approaches to Automatic Grading

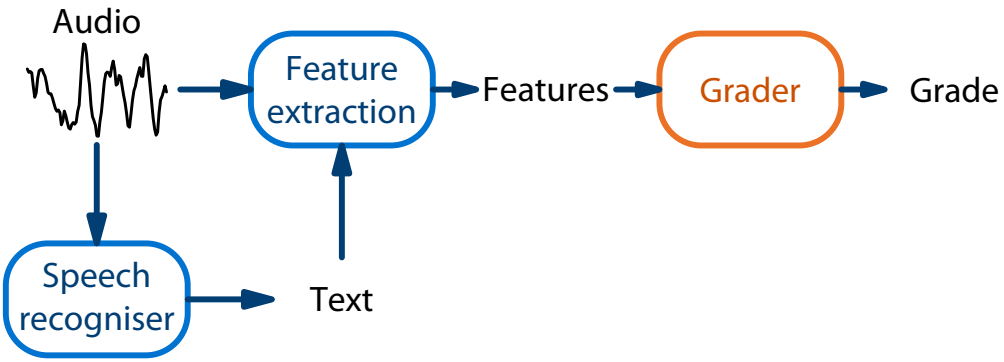


Figure 7.1 CUED Automatic Spoken Language Assessment Pipeline. This chapter will focus on the grader component.

As stated in chapter 6, automatic assessment systems, shown in figure 7.1, operate on fluency and audio features derived from automatic speech recognition (ASR) transcriptions and audio of the candidates' responses, respectively. An automatic grader is trained on these features with grade-targets provided by human examiners. Automatic grading can be treated both as a classification, where the model predicts a CEFR grade-level for each candidate, and as a regression task, where the model predicts a real value 'score' which can then be mapped to a CEFR grade. However, in this thesis automatic grading will only be investigated as a regression task due to operational requirements of Cambridge English Language Assessment, who are interested in obtaining a score which can then, if necessary, be mapped to a CEFR grade level.

As stated previously, the goal of an automatic grader to provide scores with an accuracy of a human examiner. Furthermore, it is highly desirable for an automatic grader to provide estimates of uncertainty in its predictions. An example of such a grader which has been evaluated on data from the BULATS exam is a Gaussian Process grader [119]. The estimates of uncertainty provided by a Gaussian Process can be used to, for example, back-off to human examiners [119, 80] or derive bounds on the error in the prediction. However, while Gaussian Processes are excellent for small datasets, they are difficult to scale to large datasets with

high-dimensional feature spaces due to their high computational expense [92]. Specifically, training and inference with a standard Gaussian Process are $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$ due to matrix inversion and matrix multiplication, respectively [92], where N is the size of the training data. Furthermore, the memory requirements of Gaussian Processes also have $\mathcal{O}(N^2)$ scaling. Although there is a range of approaches which can reduce the computational and memory requirements of Gaussian Processes, such as inducing points [103], sparse Gaussian Processes still require significant computational resources.

Instead of Gaussian Processes, it is possible to use deep neural networks (DNN), which can be computationally much cheaper. Unlike Gaussian Processes, which are non-parametric models, DNNs are parametric models and easily scale to very large datasets. Though they are not the most common neural network based model for regression, in this chapter *Density Networks* are considered, because unlike non probabilistic regression models, they provides estimates of uncertainty in their predictions. However, as discussed in chapter 3, standard Density Networks trained using maximum likelihood only provide measures of *data uncertainty*, or uncertainty which arises due to noise in the data, and do not capture *knowledge uncertainty*, or uncertainty arising due to mismatch between the training and test data. To be able to obtain estimates of both *data* and *knowledge uncertainty* it is necessary to either construct ensembles of Density Networks or to train them in a multi-task fashion on training OOD data, in order to explicitly build in an understanding of the limits of their knowledge. In this chapter the construction of automatic graders using standard Density Networks, ensembles of Density Networks and Density Networks trained in a multi-task fashion is investigated. These models are compared to a Gaussian processed based grader as a baseline. The rest of this section describes Gaussian Processes and Density Networks models and compares the measures of uncertainty that both types of models provide.

7.1.1 Gaussian Processes

A Gaussian Process (GP) [103] is a non-parametric Bayesian model. Unlike parametric Bayesian models, where a prior $\mathtt{p}(\boldsymbol{\theta})$ is used to derive a posterior $\mathtt{p}(\boldsymbol{\theta}|\mathcal{D})$ over *model parameters*, a Gaussian Process can be used to specify a prior and derive a posterior distribution over *functions* $f(\boldsymbol{x})$. Although it seems difficult to represent a distribution over functions, it is only necessary to define a distribution over a set of function values $\boldsymbol{f} = \{f(\boldsymbol{x}_1), \cdots, f(\boldsymbol{x}_N)\}$ at a finite, but arbitrary, set of points $\boldsymbol{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\}$. A Gaussian Process, therefore, is a joint Gaussian distribution over the values of functions $\boldsymbol{f}$ at the points $\boldsymbol{X}$:

$$\boldsymbol{f} \sim \mathcal{N}\left(m(\boldsymbol{X}), \boldsymbol{K}\left(\boldsymbol{X}, \boldsymbol{X}\right)\right) \tag{7.1}$$

where $m(\boldsymbol{X})$ is the mean and $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$ the covariance matrix as functions of $\boldsymbol{X}$. The observed outputs $\boldsymbol{y} = \{y_1, \cdots, y_N\}$[1] are assumed to be Gaussian distributed around the real function values $\boldsymbol{f}$ with homoscedastic (independent of the input) additive Gaussian noise $\mathcal{N}(0, \sigma^2)$:

$$\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{f}, \sigma^2 \boldsymbol{I}) \tag{7.2}$$

The distribution of observations $\boldsymbol{y}$ as a function of $\boldsymbol{X}$ is jointly Gaussian distributed a follows:

$$\boldsymbol{y} \sim \mathcal{N}\left(m(\boldsymbol{X}), \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I}\right) \tag{7.3}$$

where $\boldsymbol{I}$ is the identity matrix. Sampling $\boldsymbol{y}$ from this distribution correspond to sampling values of functions from the *prior* distribution over functions.

A Gaussian Process is fully specified by its mean $m(\boldsymbol{x})$ and covariance functions $k(\boldsymbol{x}, \boldsymbol{x}')$. The covariance matrix $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$ is composed of pairwise application of the covariance function or *kernel* $k(\boldsymbol{x}, \boldsymbol{x}')$, a follows:

$$\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) = \begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_1, \boldsymbol{x}_N) & \cdots & k(\boldsymbol{x}_N, \boldsymbol{x}_N) \end{bmatrix} \tag{7.4}$$

The mean function is typically chosen to be $\mathbf{0}$, as it is difficult to a priori know what is an appropriate mean function, and in practice a Gaussian Process is flexible enough to model the mean arbitrarily well [92]. A number of covariance functions can be selected, as described in [103]. In this work radial basis covariance function (RBF):

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma_y^2 \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{x}'||^2}{2l^2}\right), \tag{7.5}$$

is used as the covariance function. The shape of the RBF is parameterised by two parameters, namely l and σ_y^2. l is the length scale, which controls the how the distance between $\boldsymbol{x}$ and $\boldsymbol{x}'$ influences the covariance. σ_y^2 is the pre-set output variance which determines the average distance of the function away from its mean.

In order to make prediction for new values of $\boldsymbol{x}^*$ given the training data $\mathcal{D} = \{\boldsymbol{y}, \boldsymbol{X}\}$, a joint distribution over both the training and test targets is defined:

$$\begin{bmatrix} \boldsymbol{y} \\ y^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I} & \boldsymbol{k}(\boldsymbol{x}^*, \boldsymbol{X}) \\ \boldsymbol{k}(\boldsymbol{x}^*, \boldsymbol{X})^{\mathrm{T}} & k(\boldsymbol{x}^*, \boldsymbol{x}^*) + \sigma^2 \end{bmatrix}\right) \tag{7.6}$$

[1] In this chapter all the derivations are for a 1-dimensional real-valued output y

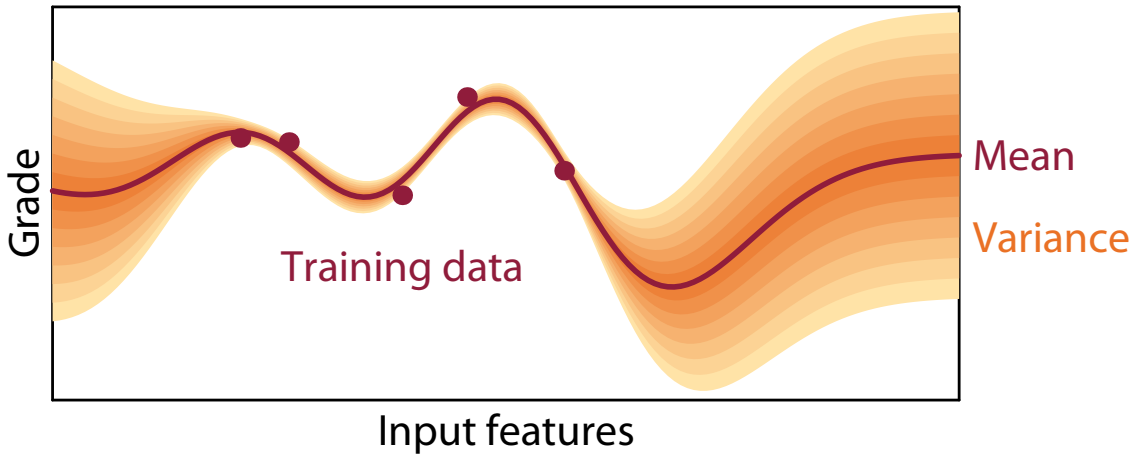


Figure 7.2 A Gaussian process trained on a few data points. The mean and variance contours are indicated. When the test point is further away from the training data, the predicted mean and variance revert to the prior.

The function $\boldsymbol{k}(\boldsymbol{x}^*, \boldsymbol{X})$ consist of the following elements by applying the covariance function $k(\cdot\,, \cdot\,)$ to the inputs:

$$\boldsymbol{k}(\boldsymbol{x}^*, \boldsymbol{X}) = \begin{bmatrix} k(\boldsymbol{x}^*, \boldsymbol{x}_1) \\ \vdots \\ k(\boldsymbol{x}^*, \boldsymbol{x}_N) \end{bmatrix} \tag{7.7}$$

The predictive distribution over y^* is obtained by conditioning on the training data $\mathcal{D}$ and the test input $\boldsymbol{x}^*$. Because $\boldsymbol{y}$ and y^* are jointly Gaussian distributed as given in (7.6), the conditional distribution is also Gaussian [11]:

$$\mathrm{p}(y^* | \boldsymbol{x}^*, \mathcal{D}) = \mathcal{N}(y^*; \hat{\mu}, \hat{\sigma}^2) \tag{7.8}$$

where the predictive mean and variance are given by the following expression:

$$\begin{aligned} \hat{\mu} &= \boldsymbol{k}(\boldsymbol{x}^*, \boldsymbol{X})^{\mathsf{T}} (\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{y} \\ \hat{\sigma}^2 &= k(\boldsymbol{x}^*, \boldsymbol{x}^*) + \sigma^2 - \boldsymbol{k}(\boldsymbol{x}^*, \boldsymbol{X})^{\mathsf{T}} (\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{k}(\boldsymbol{x}^*, \boldsymbol{X}) \end{aligned} \tag{7.9}$$

Sampling values of y^* from $\mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$ corresponds to sampling from the *posterior* distribution over functions given the data.

Figure 7.2 illustrates a toy 1-dimensional Gaussian Process trained on five data points. The horizontal and vertical axes represent the input and target values, respectively. The bands show the predicted Gaussian distribution for any input point. The middle line indicates the mean, and the coloured band the variance contours at $\frac{1}{2}$, 1 and 2 times the variance around

the means. The predictions have a low variance when close to data points and the mean interpolates between the points and to some degree extrapolates beyond. The data is assumed to be observed with noise, so the mean does not exactly go through the training points. When the prediction is requested for points further away from the training data points, the predicted distribution increases in variance. The predicted Gaussian will revert to the prior probability distribution, as when there are no training data points in the vicinity of the test point there is little to base a prediction on leading to greater uncertainty.

The predictive variance is a measure of *model uncertainty* plus a constant homoscedastic uncertainty term. As described in chapter 3 section 3.4, given an appropriate choice of prior, estimates of *model uncertainty* will capture *knowledge uncertainty*. In a Gaussian Process, the prior is chosen via choice of covariance function. Given an appropriate choice of covariance function, the predictive variance will increase the further the input $\boldsymbol{x}^*$ is away from the training data, as shown in figure 7.2. This represents the model having higher *knowledge uncertainty* in its predictions further away from the training data, where it has to do a lot of extrapolation. Note, that unlike for parametric Bayesian methods discussed in section 3.4, it is easier to obtain good estimates of *knowledge uncertainty* using non-parametric models, such as Gaussian Processes, which can explicitly measure the distance of the test input $\boldsymbol{x}^*$ from the training inputs $\boldsymbol{X}$.

7.1.2 Density Networks

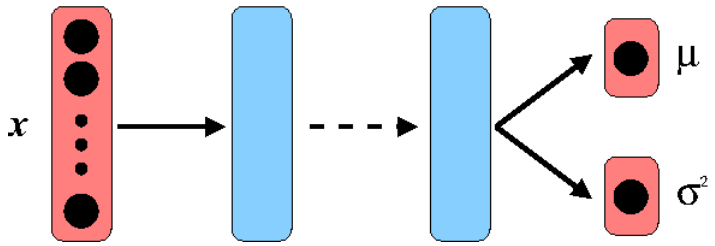


Figure 7.3 Density Network which parameterizes univariate normal distribution.

As discussed in section 2.2.2 standard parametric regression models, called regressors, are non-probabilistic, unlike standard classification models, and directly predict the target value $\hat{y}$ without yielding any estimates of uncertainty:

$$\hat{y} = f(\boldsymbol{x}^*; \hat{\boldsymbol{\theta}}) \tag{7.10}$$

However, as we have a need to obtain measures of uncertainty in predictions, this thesis has instead considered a class of probabilistic regression models called *Density Networks*, which

parameterize a continuous output density function over the targets y. Furthermore, it was shown in section 2.2.2 that regressors are actually a special case of Density Networks which predict a constant, fixed variance.

Given an appropriate choice of output density function and sufficient training data, a Density Network trained via maximum likelihood can capture *data uncertainty* - uncertainty which, for regression tasks, arises due to additive homoscedastic (input independent) and heteroscedastic (input dependent) noise. While it is possible to consider a wide range of different output distributions [26], such as the Normal, Student's T, Generalized Normal or Beta distributions, in here we only consider Density Networks which parameterize a univariate Normal distribution (figure 7.3b):

$$\begin{aligned} \mathrm{p}(y|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}}) &= \mathcal{N}(y|\hat{\mu},\hat{\sigma}) \\ \{\hat{\mu},\hat{\sigma}\} &= f(\boldsymbol{x}^*;\hat{\boldsymbol{\theta}}) \end{aligned} \tag{7.11}$$

Given a Density Network train using maximum likelihood, the differential entropy (eqn 3.21) will be the model's estimate of heteroscedastic and homoscedastic *data uncertainty*. However, we are only considering a univariate normal distribution, differential entropy is a monotonic function of the predicted variance $\hat{\sigma}^2$, which is why in this chapter the variance will be used a the model's estimate of uncertainty. Note that, while both the Gaussian Process and Density networks assume a univariate Gaussian distribution over the target y, the predictive variance of the Gaussian Process will be an estimate of *model uncertainty* plus homoscedastic *data uncertainty*, rather than heteroscedastic *data uncertainty*. Given an appropriate choice of covariance function for the Gaussian Process, the predictive variance will be an estimate of *knowledge uncertainty*. However, maximum likelihood estimation does not contain any mechanism for a model to capture *knowledge uncertainty* - uncertainty due to mismatch of the training and test distributions. Density Networks can be constructed to capture *knowledge uncertainty* in two ways, both of which are considered in this work.

Firstly, Density Networks can be trained in a multi-task fashion to yield a low entropy distribution $\mathrm{p}(y|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})$ in domain and a high entropy output distribution further away from training data. This requires out-of-distribution (OOD) training data which is used to *explicitly* build in the knowledge about the limits of the model's understanding. This OOD training data can either be obtained by using a different dataset or it can be synthesized using a generative model. In this chapter we use a factor analysis model trained on the in-domain training data to synthesize out-of-distribution training data for Density Networks. To sample features which are near the edge of the training data region the variance of the latent variable

distribution and the data covariance were both multiplied by a factor λ:

$$\begin{aligned} \boldsymbol{z} &\sim \mathcal{N}(\mathbf{0}, \lambda \cdot \boldsymbol{I}) \\ \boldsymbol{x} &\sim \mathcal{N}(\boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu}, \lambda \cdot \boldsymbol{\Sigma}) \end{aligned} \tag{7.12}$$

Density Networks can be trained using several multi-task loss functions, as discussed in section 3.3.2. In this work, based on [80], we consider the loss function which minimizes the KL-divergence between the model and a 'teacher' distribution $\mathtt{p}(y|\boldsymbol{x}; \tilde{\boldsymbol{\theta}})$ for in-domain data and the KL-divergence between the model and a target high-entropy normal distribution $\mathtt{p}_{\mathtt{out}}(y|\boldsymbol{x})$ for out-of-distribution data:

$$\mathcal{L}(\boldsymbol{\theta}) = \underbrace{\mathbb{E}_{\mathtt{p}_{\mathtt{tr}}(y,\boldsymbol{x})}\big[\mathtt{KL}[\mathtt{p}(y|\boldsymbol{x}; \tilde{\boldsymbol{\theta}})||\mathtt{p}(y|\boldsymbol{x}; \boldsymbol{\theta})]\big]}_{In-Domain\ Loss} + \gamma \cdot \underbrace{\mathbb{E}_{\mathtt{p}_{\mathtt{out}}(\boldsymbol{x})}\big[\mathtt{KL}[\mathtt{p}_{\mathtt{out}}(y|\boldsymbol{x})||\mathtt{p}(y|\boldsymbol{x}; \boldsymbol{\theta})]\big]}_{Out-of-Distribution\ Loss} \tag{7.13}$$

A standard Density Network trained using maximum likelihood will be used as the target in-domain distribution $\mathtt{p}(y|\boldsymbol{x}; \tilde{\boldsymbol{\theta}})$ while the target OOD distribution $\mathtt{p}_{\mathtt{out}}(y|\boldsymbol{x})$ is constructed by the manually setting the target OOD mean μ_{OOD} and variance σ^2_{OOD}. Since we are not interested in the model's predictions for OOD data, μ_{OOD} is set to be whatever the model predicts for OOD data ($\hat{\mu}$). The target variance σ^2_{OOD} for the OOD data is chosen to be a function of the distance of the sampled latent variable $\boldsymbol{z}$ from the mean of the latent space (0):

$$\sigma^2_{OOD} = d_{sigma} \cdot ||\boldsymbol{z}||_2 + d_{bias} \tag{7.14}$$

where d_{sigma} and d_{bias} are hyper-parameters. Given a density network trained in this fashion the predicted variance $\hat{\sigma}^2$ will be an estimate of *total uncertainty* - the sum of *data uncertainty* and *knowledge uncertainty*. In this case, a Density Network essentially *emulates* the behaviour of a Gaussian Process.

The second approach to modelling *knowledge uncertainty* using Density Networks is to construct an ensemble of M density networks $\{\mathtt{p}(y|\boldsymbol{x}^*; \boldsymbol{\theta}^{(m)})\}_{m=1}^{M}$ which yields consistent predictions in-domain and diverse predictions out-of-distribution, as described in section 3.4. In this chapter ensembles are constructed via Monte-Carlo Dropout and training M Density Networks via maximum likelihood starting from different random initializations. These are the same approaches as considered in chapter 5.

Given an ensemble of Density Networks which parameterize a univariate Gaussian distribution, it is possible to obtain estimates of uncertainty via measures of spread of the ensemble, such as the *total variance* (equation 3.36). The expression for the total variance of an ensemble of Density Networks which parameterize univariate normal distributions is

given below:

$$\begin{aligned}\underbrace{\mathbb{V}[y|\boldsymbol{x}^*]}_{Total\ Uncertainty} &= \underbrace{\mathbb{V}_{\mathrm{p}(\boldsymbol{\theta}|\mathcal{D})}\left[\mathbb{E}_{\mathrm{p}(y|\boldsymbol{x}^*,\boldsymbol{\theta})}[y]\right]}_{Model\ Uncertainty} + \underbrace{\mathbb{E}_{\mathrm{p}(\boldsymbol{\theta}|\mathcal{D})}\left[\mathbb{V}_{\mathrm{p}(y|\boldsymbol{x}^*,\boldsymbol{\theta})}[y]\right]}_{Expected\ Data\ Uncertainty} \\ &= \frac{1}{M}\sum_{m=1}^{M}\left(\hat{\mu}^{(m)} - \left(\frac{1}{M}\sum_{l=1}^{M}\hat{\mu}^{(l)}\right)^2\right)^2 + \frac{1}{M}\sum_{m=1}^{M}(\hat{\sigma}^{(m)})^2\end{aligned} \tag{7.15}$$

The expression for total variance allows *total uncertainty* to be decomposed into *data* and *knowledge uncertainty*. The first term in equation 7.15 is the variance of the mean, which captures *knowledge uncertainty*, and the second term is the mean variance, which captures the average *data uncertainty* of each member of the ensemble. Alternatively, it is possible to calculate expected pairwise KL-divergence (EPKL) between each member of the ensemble (equation 3.35). As discussed in section 3.4, the EPKL is an upper bound of the mutual information and is a measure of *knowledge uncertainty*.

Finally, it is possible to combine single-model and ensemble approaches, and consider explicit ensembles of Density Networks trained in a multi-task fashion from different random initializations. Measures of uncertainty can be derived from them in the same way as from an ensemble of standard Density Networks.

7.2 Experimental Evaluation

The previous sections discussed construction of automatic graders using both Gaussian Processes, the baseline grader considered in this chapter, and Density Networks. In this section these approaches are evaluated on data from the BULATS [18, 15] spoken language proficiency exam provided by Cambridge English Language Assessment. Models are evaluated on their ability to assign accurate grades, their ability to reject and pass over a candidate for assessment by a human examiner based on their uncertainty estimates and on the quality of the predictive intervals which the models provide. The evaluation metrics are discussed in subsection 7.2.1, the details of the datasets are described in subsection 7.2.2 and the details of model construction are found in subsection 7.2.3. Evaluation of predictive performance, use of uncertainty estimates for prediction rejection and derivation of predictive intervals are discussed in sections 7.2.4-7.2.6.

7.2.1 Assessment Criteria

In this set of experiments, the automatic graders will be assessed using several metrics of performance. Firstly, given a test set $\mathcal{D}_{test} = \{\boldsymbol{x}_i^*, y_i^*\}_{i=1}^N$ an automatic grader's predictions

$\{\hat{\mu}_i\}_{i=1}^N$ will be assessed via mean square error (MSE) and mean absolute error (MAE):

$$\begin{aligned}\mathcal{L}^{\texttt{MSE}}(\boldsymbol{\theta}, \mathcal{D}_{test}) &= \frac{1}{N}\sum_{i=1}^{N}(\hat{\mu}_i - y_i^*)^2 \\ \mathcal{L}^{\texttt{MAE}}(\boldsymbol{\theta}, \mathcal{D}_{test}) &= \frac{1}{N}\sum_{i=1}^{N}|\hat{\mu}_i - y_i^*|\end{aligned} \tag{7.16}$$

which measure the average square deviation and the average absolute deviation of the predictions from the targets, respectively. The difference between square and absolute error is that the former is more sensitive to outliers, while the latter is more robust to them. Another measure of predictive performance is the Pearson correlation coefficient ρ [92]:

$$\begin{aligned}\rho &= \frac{\sum_{i=1}^{N}(\hat{\mu}_i - \bar{\mu})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(\hat{\mu}_i - \bar{\mu})^2)}\sqrt{\sum_{i=1}^{N}(y_i - \bar{y})^2}} \\ \bar{y} &= \frac{1}{N}\sum_{i=1}^{N} y_i, \quad \bar{\mu} = \frac{1}{N}\sum_{i=1}^{N}\hat{\mu}_i\end{aligned} \tag{7.17}$$

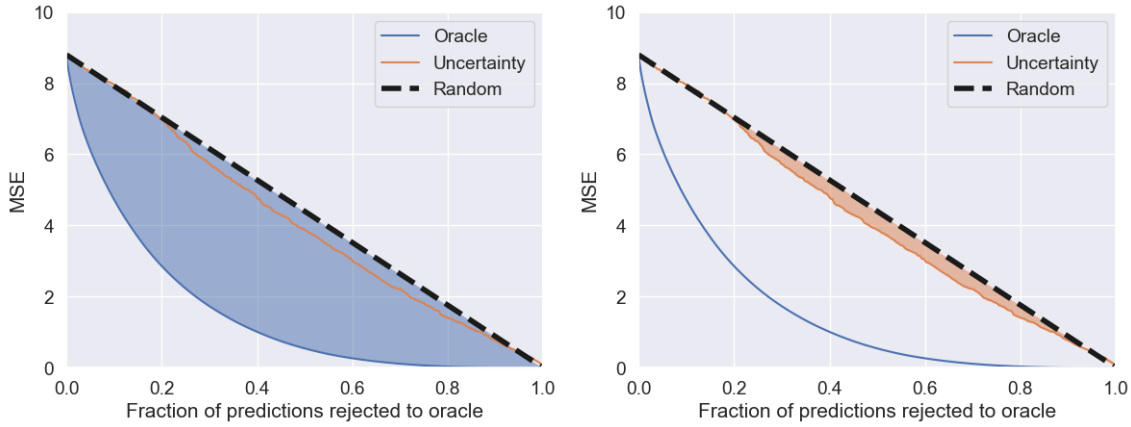
where $\bar{y}$ is the target sample mean and $\bar{\mu}$ is the prediction sample mean. Pearson correlation coefficient measures how strong a *linear* relationship exists between the predictions and the targets. A $\rho = 1$ implies there is a perfectly linear correlation and $\rho = -1$ means there is a perfectly linear inverse correlation. A $\rho = 0$ means that the data is not *linearly* correlated. However, the data may have a complex, non-linear relationship, which will not be reflected in a PCC[2] value. Unlike MSE and MAE, which give the average square or absolute deviation, PCC assesses whether there is a linear relationship between the predictions and the targets. A situation were PCC is very high, for example 99.0, while MSE or MAE are non zero, implies that MSE and MAE can be reduced to zero by using an appropriate affine transformation.

An operational requirement of Cambridge English is that 50% of the predictions must fall within half and 90% must fall within a single CEFR [96] grade level. In terms of BULATS scores which range 0-30, this means that 50% of the predictions must have an absolute error less than 2.5, and 90% of the predictions an absolute error less than 5.0, according to the BULATS-CEFR mapping in table 6.2.

Having discussed how to assess predictive performance, let's now consider how to assess the quality of uncertainty estimates. Unlike classification tasks, where it is possible to assess misclassification using Precision-Recall curves, the same is not possible for regression tasks, as the prediction are real-valued. However, it is possible to use prediction rejection curves,

[2] In this work, $\rho \cdot 100$ is reported in order to keep tables compact.

introduced in chapter 5, to assess whether the uncertainty correlates well with the MSE/MAE. As stated in the beginning of this chapter, the operating scenario of interest is to use an automatic grader's estimates of the uncertainty to reject and pass over candidates to be assessed by human graders for high-stakes tests. The goal is to maximize the increase in average performance on the remaining set of candidates while rejecting the least number of candidates. This represents the same scenario as the rejection curves. The rejection curves for regression are shown in figure 7.4. As the fraction of predictions rejected is increased, model predictions are replaced with oracle (human) scores in some particular order, decreasing the MSE. Figure 7.4 depicts 3 curves representing different orders of rejection: expected random



(a) Shaded area is $\mathtt{AR}_{\mathtt{orc}}$

(b) Shaded area is $\mathtt{AR}_{\mathtt{uns}}$

Figure 7.4 Example prediction rejection curves for regression.

rejection order, oracle rejection order and uncertainty-based rejection order. The expected random performance curve is a straight line from the base predictive performance (MSE) to 0, representing rejection in a random order. The oracle rejection curve is constructed by rejecting predictions in order of decreasing mean square error relative to human graders. A rejection curve generated by rejecting candidates in order of decreasing uncertainty should sit between the random and oracle curves.

As before, the rejection curve can be summarized using the rejection ratio $\mathtt{RR}$ (eqn. 7.18), which is the ratio of the areas between the uncertainty-based ($\mathtt{AR}_{\mathtt{uns}}$) and oracle ($\mathtt{AR}_{\mathtt{orc}}$) rejection curves relative to the random rejection curve. Ratios of 1.0 and 0.0 correspond to perfect and random rejection, respectively. A rejection ratio less than 0 indicates the 'perverse' situation where reported uncertainty is greater for candidates with smaller actual error.

$$\mathtt{RR} = \frac{\mathtt{AR}_{\mathtt{uns}}}{\mathtt{AR}_{\mathtt{orc}}} \tag{7.18}$$

However, there is a limitation to rejection curves when applied to regression tasks. Using bias-variance decomposition [92], it is possible to show that MSE is composed of reducible error (given by bias and variance), and heteroscedastic or homoscedastic target noise:

$$\mathbb{E}_{\mathtt{p_{tr}}(\boldsymbol{x},y)}\left[(y-\hat{\mu})^2\right] = \underbrace{\mathbb{E}_{\mathtt{p_{tr}}(\boldsymbol{x})}\left[(\hat{\mu}-\mathbb{E}_{\mathtt{p_{tr}}(y|\boldsymbol{x})}[y])^2\right]}_{Reducible\ Error} + \underbrace{\mathbb{E}_{\mathtt{p_{tr}}(\boldsymbol{x})}\left[\mathbb{V}_{\mathtt{p_{tr}}(y|\boldsymbol{x})}[y]\right]}_{Irreducible\ Error} \tag{7.19}$$

Even if systematic error is eliminated there will still be random errors due to heteroscedastic or homoscedastic noise on the targets. This means that when a model is very high-performing, then MSE will be composed only of random error and the oracle ordering will be completely determined by the noise. If the noise is homoscedastic, then oracle rank-ordering is in fact random rank ordering. However, as an 'oracle' rejection curve can still be drawn based on the actual random errors made, it is possible to obtain a very low $\mathtt{AUC_{RR}}$ in this situation. This means that $\mathtt{AUC_{RR}}$ numbers for high performance models are meaningless when there is only homoscedastic noise, as the oracle rejection curve is an over-estimation. On the other hand, if the noise on the targets is heteroscedastic, then it may be possible to obtain a rank ordering, but the oracle ordering will still be over-estimated to some degree, as it is only noise which determines the rank-order between two points who have equal heteroscedastic uncertainty.

Another approach to assessing the quality of uncertainty estimates which does not depend on rank ordering the predictions is via *calibration*, which assesses how well the errors made are captured by the model's *predictive intervals* δ. Consider a model which parameterizes a univariate normal distribution and yields a set of means and variances $\{\hat{\mu}_i, \hat{\sigma}_i^2\}_{i=1}^N$. The goal is to assess what fraction of the predictions $\hat{\mu}_i$ are within a certain predictive interval δ of the targets y_i. Using the variance $\hat{\sigma}_i^2$ the theoretical fraction $\mathtt{f_t}$ of predictions within a interval δ of the target can be calculated via the cumulative distribution function (CDF) F:

$$\begin{aligned} \mathtt{f_t} &= \mathtt{P}(\hat{\mu}-\delta \leq y < \hat{\mu}+\delta; \hat{\sigma}) \\ &= F(\frac{\hat{\mu}+\delta}{\hat{\sigma}}) - F(\frac{\hat{\mu}-\delta}{\hat{\sigma}}) \end{aligned} \tag{7.20}$$

In the case of a normal output density function 68% of the predictions $\hat{\mu}_i$ should be within 1 standard deviation $\hat{\sigma}_i$ of the targets y_i, 95% within 2 standard deviations, etc... Using the inverse CDF, a predictive interval δ_i can be calculated as a function of the desired theoretical fraction and the predicted variances for each prediction:

$$\delta_i = \hat{\sigma}_i \cdot F^{-1}(1-\frac{\mathtt{f_t}}{2}) - \hat{\mu}_i \tag{7.21}$$

It is then possible to calculate the *empirical fraction* $\mathtt{f_e}$ of predictions within the predictive intervals:

$$\mathtt{f_e} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{I}(|\mu_i - y_i| \leq \delta_i) \tag{7.22}$$

The theoretical and empirical fractions can then be plotted on a *calibration curve* as the theoretical fraction goes form 0.0 to 1.0, as shown in figure 7.5. A calibration curve assesses whether the empirical fraction $\mathtt{f_e}$ of the predictions within a certain interval δ matches the probability $\mathtt{f_t}$ given by cumulative density function at that interval. If the curve is above the line, then the model is *under confident* and its predictive intervals are bigger than the errors made, and if the curve is below the line, the model is *over confident* and yields predictive intervals which are smaller than the errors made. A well calibrated model should yield predictive intervals such that the empirical and theoretical fractions are equal. This means that the model captures the uncertainty in the prediction well. Note, calibration is an average across a dataset, with predictions on certain subsets of the data being better or worse calibrated.

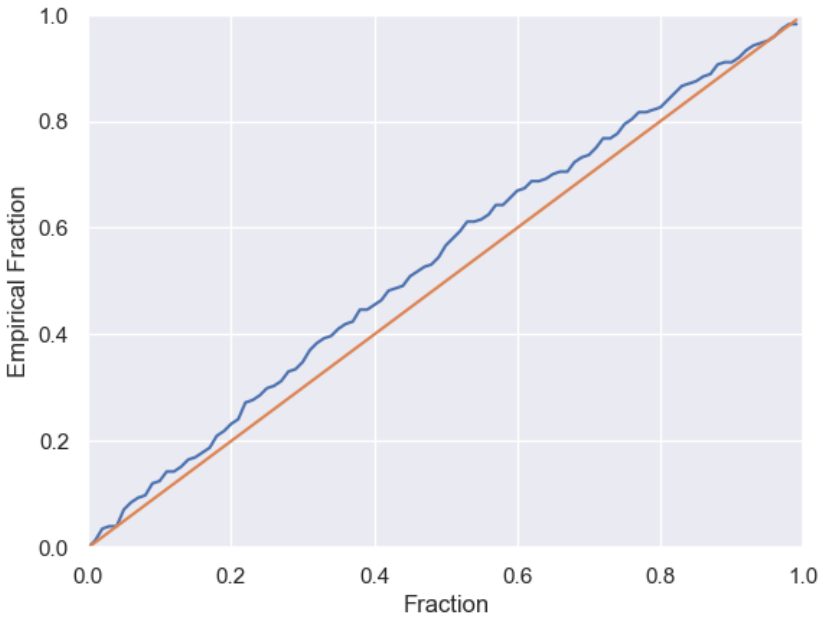


Figure 7.5 Calibration Curves

7.2.2 Datasets

In this chapter automatic graders are trained on datasets derived form real candidates' responses to questions on the BULATS spoken language proficiency exam provided by

Dataset	Num.Spks	ASR % WER
BLT-TRAIN	4299	-
BLT-EVAL	224	48.6

Table 7.1 Description of datasets in terms of number of training samples and ASR word error rate (% WER) relative to crowdsource transcriptions.

Cambridge English Language Assessment. As described in section 6.1, the BULATS exam has a simple question section, a read-aloud section and three sections which elicit spontaneous, unstructured responses to open-ended questions. In this work one training dataset BLT-TRAIN and one evaluation dataset BLT-EVAL, described in table 7.1, are considered. BLT-TRAIN contains grade-targets provided by 'standard' BULATS examiners, while BLT-EVAL contains grade targets provided by 'expert' examiners, who train the standard examiners. However, the effect of this mismatch is likely to be minimal, as the primary difference between grade-targets assigned by standard and expert examiners is the level of noise, rather than a systematic shift in assignment of grades.

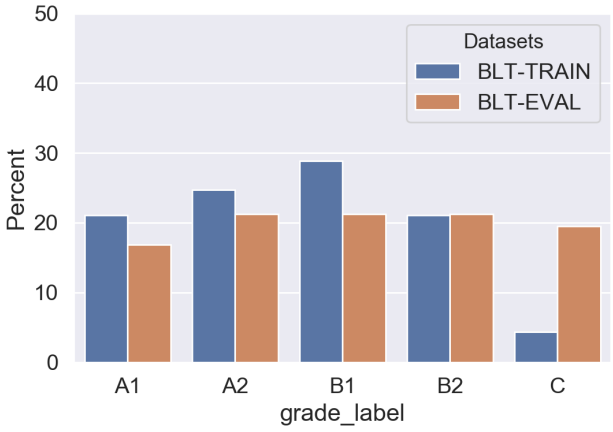


Figure 7.6 CEFR grade distribution of datasets. Here grades C1 and C2 are combined into one, because there are so few C2 speakers.

A set of 10 audio features and 23 fluency features, described in chapter 6 table 6.3, is derived from all responses of a candidate's responses to questions on the BULATS. The ASR system has a word error rate (WER) of 48.6 on the BLT-EVAL dataset relative to crowd-sourced transcriptions [120]. Despite this high word error rate, previous work [119] has shown that the current feature set is not sensitive to the ASR performance. As shown in figure 7.6, the evaluation dataset has a roughly equal distribution across all grade levels. The training dataset BLT-TRAIN is also roughly balanced across the all grade levels except C, which only makes up 5% of the dataset. Figure 7.7 details the L1 breakdown of each

dataset. BLT-TRAIN and BLT-EVAL contain candidates from the sames L1 languages, except Spanish. However, while BLT-EVAL is evenly distributed across L1 languages, BLT-TRAIN is biased towards Thai, Spanish and Arabic speakers.

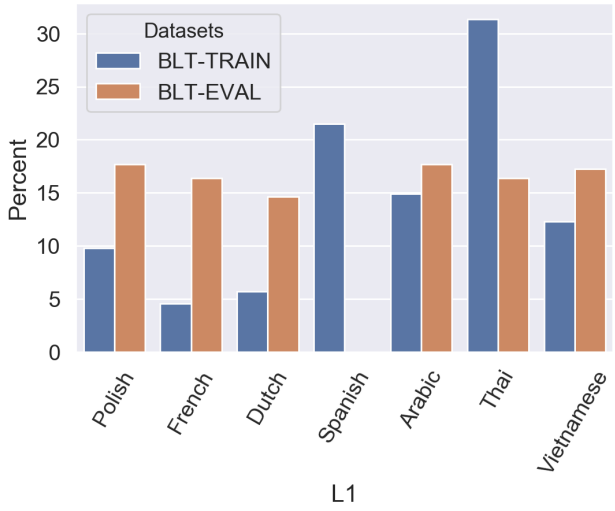


Figure 7.7 L1 language distribution of datasets. Note, there are no Spanish L1 candidates in the evaluation set.

7.2.3 Model Details

In the following experiments Gaussian Process and a range of Density Network graders are constructed on the BLT-TRAIN dataset. For all models, the input features are whitened by subtracting the mean and dividing by the standard deviation of each feature computed on the training data. The Gaussian Process (GP) uses an RBF kernel with homoscedastic Gaussian noise and is trained using the scikit-learn [100] version 17.0 Gaussian Process implementation.

Several types of Density Networks are trained in the following experiments. Firstly, standard Density Networks (DDN) are trained via maximum likelihood. The Density Networks with 2 hidden layers with 180 Leaky ReLU units in each layer were trained using the ADAM [62] optimizer. An exponentially decaying learning rate with decay factor of 0.95 per epoch and an initial learning rate of 1e-2 was used. Dropout [117] regularization was used for training with a keep probability of 0.8. Additionally, L2 regularization was used with a weight of 4e-7. Models were trained for 100 epochs.

To evaluate sensitivity to random initialization, 10 density networks were constructed using different random seeds. These 10 Density Networks were also evaluated as an ensemble (ENS). Additionally, a Monte Carlo Dropout Ensemble (MCDP) was constructed from each

Density Network using test-time dropout on each Density Network with a keep probability of 0.8.

Density Networks trained in a multi-task fashion (DDN-MT) on in-domain and out-of-distribution data were also constructed. 10 networks were constructed evaluated sensitivity to initialization. Each DDN-MT model was initialized from a standard DDN model and trained for a further 100 epochs with a learning rate of 1e-4 using OOD data generated via factor analysis. The predictions of the standard DDN models were used as the targets of the DDN-MT models. Out-of-distribution training data was synthesized using a factor analysis model with a 10-dimensional latent space and hyper-parameters $\lambda = 1.1,\ d_{sigma} = 6,\ d_{bias} = 20$. These hyper parameters were chosen such that the target OOD variances are of the same order but larger than the variance of the real data. Density Networks trained in a multi-task fashion were also evaluated jointly as an ensemble (ENS-MT).

7.2.4 Evaluation of Predictive Performance

In the current subsection the predictive performance of all models is evaluated using MSE, MAE and PCC as well as percentage of predictions within one-half and one-whole CEFR-grade deviation. Results are presented in table tables 7.2, which shows that all models comparable performance, with the ensemble of 10 multi-task trained Density Networks (ENS-MT) achieving the best results, while ensembles generated via Monte-Carlo dropout yield the worst predictive performance in term of MSE and MAE. All models achieve a performance of 87.0 % PCC or above. Notably, Density Networks trained in a multi-task fashion yields more accurate results than standard Density Networks in terms of MSE and MAE. All models achieve results comparable to a standard human examiner in terms of pearson correlation, as the PCC of standard examiners with expert examiners is 87.5. In terms of MSE all models achieve super-human performance, as the MSE of standard examiners relative to expert examiners is 14.20 . This shows that automatic assessment systems already better, faster and more consistent than standard humans examiners, despite the limited nature of the features.

7.2.5 Evaluation of Rejection Performance

Having established that all models are able to achieve high predictive performance, it is now necessary to assess the quality of the uncertainty estimates provided by each model. In this section, the uncertainty estimates derived on Density Networks, Density Networks trained in a multi-task fashion and Ensembles are compared to uncertainty estimates derived from a Gaussian Process on the task of prediction rejection. Table 7.3 shows the performance,

Model	GP	Single Model		Ensemble		
		DDN	DDN-MT	MCDP	ENS	ENS-MT
MSE	8.66 ± NA	8.80 ±0.26	8.66 ±0.19	8.91 ±0.25	8.62 ± NA	**8.58** ± NA
MAE	2.30 ± NA	2.32 ±0.05	2.28 ±0.03	2.34 ±0.05	2.29 ± NA	**2.27** ± NA
PCC	**87.5** ± NA	87.1 ±0.1	87.0 ±0.1	87.1 ±0.1	87.4 ± NA	87.1 ± NA
% AE < 2.5	62.1 ± NA	52.2 ±0.0	**62.6** ±0.0	53.0 ±0.0	53.1 ± NA	61.1 ± NA
% AE < 5.0	90.6 ± NA	82.2 ±0.0	91.7 ±0.0	82.2 ±0.0	81.7 ± NA	**92.0** ± NA

Table 7.2 Grading performance on BLT-EVAL. Results for DDN, DDN-MT and MCDP are means $\pm 2\sigma$ across 10 model trained from different random initializations.

in rejection ration RR, of rejection for all models using a range of uncertainty measures. The best rejection performance is achieved using an explicit ensemble of Density Networks trained in a multi-task fashion (ENS-MT), while the Gaussian Process yields the second-best results. Notable, the best measures of uncertainty for rejection are based on *knowledge uncertainty*, while rejecting based on estimates of *data uncertainty* seems to perform worse. The exception is ENS-MT, where the distinction between *knowledge uncertainty* and *data uncertainty* is not so clear, as each DDN-MT model in the ensemble yields an estimate of *total uncertainty*. Firstly, it suggests that the largest errors occur on outlier candidates who are somehow mismatched to the training data, which is why measures of *knowledge uncertainty* yield better rejection performance. In section 7.2.2 is it shown that there are several levels of mismatch between the training and evaluation data, which may be detected using measures of *knowledge uncertainty*. Secondly, it suggests that there is little *heteroscedastic* data uncertainty, and most *data uncertainty* is homoscedastic. The low rejection ratios using measures of *data uncertainty* and the large variation (indicated via $\pm 2\sigma$) across different initializations supports this assertion.

Model	GP	Single Model		Ensemble		
		DDN	DDN-MT	MCDP	ENS	ENS-MT
Total Variance	-	-	22.7 ± 5.8	9.8 ±5.0	13.1 ± NA	23.8 ± NA
Mean Variance	-	10.1 ±7.2	-	10.5 ±6.8	9.9 ± NA	23.7 ± NA
Variance of Mean	22.8 ± NA	-	-	4.7 ±6.6	22.7 ± NA	17.1 ± NA
EPKL	-	-	-	3.8 ±9.6	21.4± NA	**27.4** ± NA

Table 7.3 Prediction rejection performance using Rejection Ratio RR on BLT-EVAL dataset. Results for DDN, DDN-MT and MCDP models are mean rejection ratios $\pm 2\sigma$ across random initializations.

The rejection ratios on this task are far lower than the rejection ratios achieveable on the classification tasks considered in chapter 5. It is not clear whether this is due to the limitation of rejection curves for regression tasks (being dominated by homoscedastic noise), or due to the models failing to detect dataset shift. However, the fact that Gaussian Process, multi-task trained Density Networks and ensembles operated on different principles (non-parametric model, single parametric model, ensemble of parametric models), but achieve comparable rejection performance, suggests that the limitation lies in the features, which do not properly reflect certainty types of dataset shift. It is likely that a richer and more robust set of features may allow all models to extract better estimates of uncertainty. Alternatively, investigation of grading as a classification task may be able to answer whether it is in fact the features which are limited, or whether the errors are dominated by homoscedastic noise.

7.2.6 Evaluation of Calibration Performance

Instead of assessing whether it is possible to rank-order errors based on predicted uncertainty, it is possible to evaluate whether the error falls within predicted error bounds or *predictive interval*. This is done by constructing a calibration curves, as discussed in section 7.2.1. The calibration curves for all models are presented in figure 7.8. The calibration curves show that all models are generally well-calibrated, with DDN models yielding the most calibrated estimates of uncertainty. DDN-MT, ENS-MT and Gaussian Processes tend to produce the most under-confident predictions. This makes sense, as multi-task trained Density Networks are biased, while Gaussian Processes do not capture heteroscedastic *data uncertainty*. [3]

7.3 Chapter Summary

The current chapter discussed the application of Deep Learning and uncertainty estimation to the task of automatic grading of spoken language proficiency exams based on features derived from audio and ASR transcriptions. Automatic graders based on Gaussian Processes and Density Networks were explored. Models were evaluated on their predictive performance as well as on the quality of their uncertainty estimates.

In section 7.2.4, it was shown that the best predictive performance in terms of mean absolute and mean square deviation from the targets was the ensemble of Density Networks trained in a multi-task fashion. At the same time, the best performance in terms of Pearson correlation was demonstrated by the Gaussian Process. However, all models are able to outperform standard human examiners when assessed against expert human examiners using

[3] Though it is possible to consider heteroscedastic Gaussian Processes.

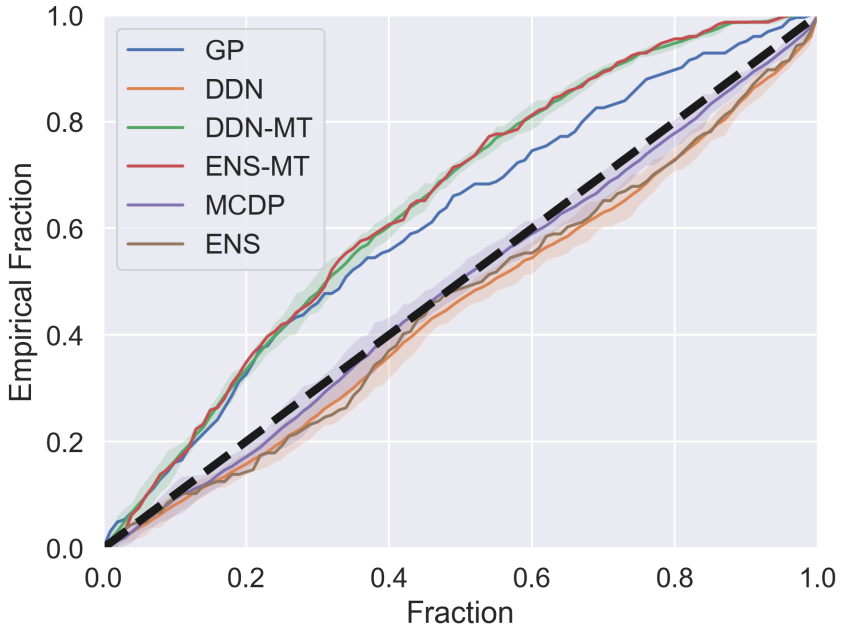


Figure 7.8 Mean calibration curves of models across 10 different random initializations (except ENS, ENS-MT and GP) with $\pm 2\sigma$ error bounds.

the metrics considered in this chapter. This shows that automatic assessment systems are already reaching human-level performance, despite the limited nature of the features and size of training data. At the same time, future work should explore new ways of assessing these models in order to reveal hidden biases and vulnerabilities. Additionally, the comparable performance of models based on Density Networks and Gaussian Processes suggests that it is possible to achieve the same predictive performance at a lower computational and memory cost. Thus, future work should explore training these models on far larger datasets.

In section 7.2.5, estimates of uncertainty in predictions provided by all models were evaluated on the task of rejecting predictions in the order of decreasing uncertainty. The results showed that standard Density Networks trained via maximum likelihood and Monte-Carlo dropout ensembles fail to meaningfully reject predictions. Explicit ensembles of Density Networks, both standard and trained in a multi-task fashion, as well as the Gaussian Process are able to successfully reject predictions. Furthermore, it was shown that rejecting based on measures of *knowledge uncertainty* yields better performance than based on measures of either *total* uncertainty or *data uncertainty*. This suggests that applying regression Prior Networks, developed in chapter 4, to this task is an interesting avenue of future research. Additionally, construction of classification models for grading should also be investigated.

Finally, uncertainty estimates were also assessed on the quality of *predictive intervals* that they yield in section 7.2.6. It was shown that explicit ensembles, Monte-Carlo dropout

ensembles and Density Networks are well calibrated, but Density Networks trained in a multi-task fashion to capture *knowledge uncertainty*, explicit ensembles derived from them and Gaussian Processes are under-confident.

Chapter 8

Deep Learning for Prompt-Response Relevance Assessment

The previous chapter examined deep learning models for constructing automatic graders for spoken language proficiency assessment. Due to the limited nature of the available features, these automatic graders primarily focus on pronunciation and fluency, both of which are highly correlated with proficiency. At the same time content assessment is minimal. However, reliable and robust assessment of proficiency requires the evaluation of the semantic content, construction and relevance of a response to the question prompt. Crucially, an automatic system must assess if the candidate's response is relevant to the exam prompt they are answering. Due to the limited nature of the features which they use, current graders are incapable of detecting when a candidate has given a non-relevant response, either maliciously, by reciting memorized English text, or non-maliciously, due to misunderstanding the question. This vulnerability compromises the reliability of an automatic assessment system and prevents deployment to high-stakes examinations.

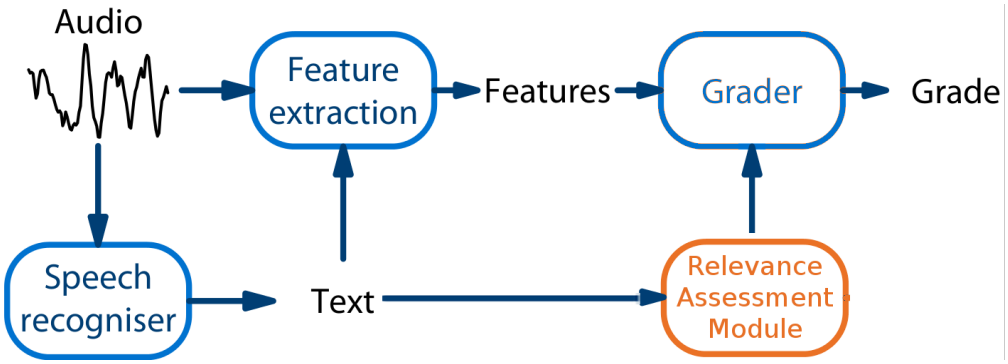


Figure 8.1 CUED Automatic Spoken Language Proficiency Pipeline with a Relevance Assessment Assessment module

This chapter seeks to develop a solution to this problem by developing a *prompt-response relevance detection module* for the automatic assessment pipeline. The goal of this module will be to assess, based on an automatic transcription of the candidate's spoken response, whether the given response is relevant to the question prompt which the candidate was asked. Two distinct types of approaches to relevance assessment are considered - an indirect approach based on prompt-topic classification, and a direct approach, where the model yields a relevance score.

If a relevance assessment module is part of an automatic assessment system deployed for high states assessment it must also provide estimates of uncertainty in its predictions, much like the automatic graders discussed in the previous chapter. Ensemble approaches to uncertainty estimation, described in chapter 3, are applied to the models developed in this chapter. The estimates of uncertainty are evaluated on the tasks of misclassification detection introduced in chapter 5.

This chapter is structured as follows: section 8.1 frames the problem of prompt-response relevance assessment in mathematical terms. Indirect relevance assessment approaches are discussed in section 8.2 and direct relevance assessment approaches are discussed in section 8.3. Indirect relevance assessment models are evaluated on prompts and responses from the BULATS exam in section 8.5. Direct relevance assessment models are evaluated on prompts and responses from the BULATS and LinguaSkill exams in section 8.5. Specifically, sections 8.5.5 and 8.5.6 investigate the predictive performance of models on evaluation data which is either matched or mismatched to the training data, respectively. The derivation of uncertainty estimates using ensemble approaches is investigated in section 8.5.7. The code and experiments for the direct models was a product of joint work with Bruno Mlodozeniec.

8.1 Prompt-Response Relevance Assessment

The current section introduces the task of prompt-response relevance assessment and defines the problem in mathematical terms. Consider the following scenario - there is an exam where candidates must provide a response $\boldsymbol{w}_r$ to an open ended question prompt $\boldsymbol{w}_p$ chosen from a pool prompts of $\mathcal{D}_p = \{\boldsymbol{w}_p^{(q)}\}_{q=1}^{Q}$ which can be asked on this exam. Here, both the response and the prompt are variable length sequences of words w:

$$\boldsymbol{w} = \{w^{(1)}, \cdots, w^{(L)}\},\ w \in \{\omega_1, \cdots, \omega_V\} \tag{8.1}$$

In the context of assessment of spoken response relevance, the response sequence is produced by a speech recognition system, while the prompt is written text. The goal of a relevance

assessment system is to assess whether the response is semantically relevant to the question asked - essentially, is the candidate answering the question-prompt, or are they speaking off topic. In this chapter two classes of approach to relevance assessment are considered.

The first approach treats relevance assessment as a 'topic' classification task. Here each prompt is considered to correspond to a distinct *class* or *topic*[1]. A prompt topic classifier $\texttt{P}(y_p = q|\boldsymbol{w}_r)$ is constructed which predicts to which prompt the response $\boldsymbol{w}_r$ relates to:

$$\hat{y}_p = \arg\max_q \{\texttt{P}(y_p = q|\boldsymbol{w}_r; \hat{\boldsymbol{\theta}})\} \tag{8.2}$$

If the predicted prompt-topic and the true prompt-topic are the same, then the response is deemed relevant, otherwise it is considered non-relevant:

$$\texttt{rel} = \begin{cases} 1, \ y_p = \hat{y}_p \\ 0, \ y_p \neq \hat{y}_p \end{cases} \tag{8.3}$$

Conceptually, this approach seeks to answer the question "Which prompt is most likely have elicited the given response?". This question is subtly different from 'is the given response relevant to the prompt?", which is answered only *indirectly*. Consequently, this approach will be referred to as an *indirect approach* to relevance assessment.

An important property of this approach is that it assumes that prompt-topics are mutually exclusive and that a response can only be relevant to a single prompt. This approach does not fully allow for a response to have *degrees of relevance* to various prompts. However, certain prompts in an exam can ask similar questions, which means that it is possible for the model to yield a large number of false-negatives. Thus, it may be beneficial to relax this mutual-exclusivity to a certain degree by considering whether the true prompt-topic is within the top k predicted prompt-topics:

$$\texttt{rel} = \begin{cases} 1, \ y_p \in \{\hat{y}_1, \cdots, \hat{y}_K\} \\ 0, \ y_p \notin \{\hat{y}_1, \cdots, \hat{y}_K\} \end{cases} \tag{8.4}$$

Another limitation of these approaches is that the training data for the prompt-topic classifier must contain examples of responses for each prompt-topic. This limits the flexibility of deployment, as example responses must be collected every time a new prompt is introduced. Furthermore, a classifier over a fixed number of classes may have to be re-trained every time a new prompt is added to the pool of prompts, which may be expensive.

[1]Here the concept of a 'topic' is used in a slightly different way than in topic models like LDA [101]

Instead of treating relevance assessment as a prompt-topic classification task, it is possible to consider *directly* assessing prompt-response relevance by constructing a model which yields a *relevance score* $\mathcal{S}(\boldsymbol{w}_r, \boldsymbol{w}_p)$ between a prompt and a response. Here, a response is considered relevant to a prompt if the relevance score is above a certainty threshold T:

$$\texttt{rel} = \begin{cases} 1, & \mathcal{S}(\boldsymbol{w}_r, \boldsymbol{w}_p) \geq T \\ 0, & \mathcal{S}(\boldsymbol{w}_r, \boldsymbol{w}_p) < T \end{cases} \tag{8.5}$$

An example of such a relevance score is the probability of relevance produced by a probabilistic discriminative model:

$$\texttt{P}(\texttt{rel}|\boldsymbol{w}_r, \boldsymbol{w}_p; \hat{\boldsymbol{\theta}}) \tag{8.6}$$

This class of approaches fully allows a response to have various degrees of relevance to multiple prompts at the same time. Furthermore, such models are not limited to assessing relevance of a response to only a fixed number of prompts. If the model is powerful enough and generalizes well, it may be able to assess relevance to new prompts which were not previously seen in the training data.

8.2 Indirect Prompt-Response Relevance Assessment

The two approaches to relevance assessment considered in this chapter were introduced in the previous section - the indirect approach, which involves classifying which prompt is most likely to have elicited the given response, and the direct approach, where a model directly yields a relevance scores. This section discusses two methods of implementing the indirect approach to prompt-response relevance assessment. Information retrieval style approaches, where a K-nearest neighbours (KNN) classifier is constructed based on distances between bag-of-words vector representations of responses and prompts, are discussed in section 8.2.1. Section 8.2.2 investigates use of a prompt conditional language model using a recurrent neural network to obtain posterior probabilities over prompts via application of Bayes' rule.

8.2.1 Vector distances based Approaches

Previous work on relevance assessment between prompts and *written* responses [132] has used information-retrieval style approaches based on evaluating distances between vector representations of written prompts and responses. In this section we considering applying similar approaches to the assessment of relevance between *spoken* responses to open ended prompts, where the response word sequence $\boldsymbol{w}_r$ is generated using a speech recognition

system. Standard information-retrieval approaches to assessing the relevance involve the use measures of semantic similarity between responses and prompts $\mathtt{D}_{\mathtt{sem}}(\boldsymbol{w}_r, \boldsymbol{w}_p)$. Prompt classification is done by finding the prompt which is closest to the response:

$$\hat{y}_p = \arg\min_q \{\mathtt{D}_{\mathtt{sem}}(\boldsymbol{w}_r, \boldsymbol{w}_p^{(q)})\} \tag{8.7}$$

The response is considered relevant if the predicted prompt topic $\hat{y}$ is equal to the actual prompt topic y, otherwise the response is considered non-relevant. As discussed in section 8.1, classification accuracy can be improved by considering whether the true prompt is in the top K closes prompts to the response. However, this does increase the rate of false-positives.

In practice, the semantic distance $\mathtt{D}_{\mathtt{sem}}$ between response $\boldsymbol{w}_r$ and prompt $\boldsymbol{w}_p$ can be approximated by considering a distance metric $\mathtt{D}_{\mathtt{vec}}$ between vector representations of the response $\boldsymbol{h}_r$ and the prompt $\boldsymbol{h}_p$.

$$\mathtt{D}_{\mathtt{sem}}(\boldsymbol{w}_r, \boldsymbol{w}_p) \approx \mathtt{D}_{\mathtt{vec}}(\boldsymbol{h}_r, \boldsymbol{h}_p) \tag{8.8}$$

A common similarity metric $\mathtt{D}_{\mathtt{vec}}(\boldsymbol{h}_r, \boldsymbol{h}_p)$ is cosine similarity, which measures the cosine of the angle between two vectors. A distance metric based on this, *cosine distance*, can be defined as:

$$\mathtt{D}_{\mathtt{cos}}(\boldsymbol{h}_r, \boldsymbol{h}_p) = 1 - \frac{\boldsymbol{h}_r \cdot \boldsymbol{h}_p}{|\boldsymbol{h}_r||\boldsymbol{h}_p|} \tag{8.9}$$

An alternative distance metric is the *Mahalanobis distance*:

$$\mathtt{D}_{\mathtt{mah}}(\boldsymbol{h}_r, \boldsymbol{h}_p) = (\boldsymbol{h}_r - \boldsymbol{h}_p)^{\mathtt{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{h}_r - \boldsymbol{h}_p) \tag{8.10}$$

where $\boldsymbol{\Sigma}^{-1}$ is the global covariance matrix derived from all vectors $\boldsymbol{h}_r$ and $\boldsymbol{h}_p$ in a dataset. If the covariance matrix is diagonal, this reduces to the *scaled euclidean distance*. In practice, cosine distance is preferred, as it is computationally cheaper to evaluate and does not require the estimation of a global covariance matrix.

Vector representations $\boldsymbol{h}_r$ and $\boldsymbol{h}_p$ the response sentence $\boldsymbol{w}_r$ and the prompt sentence $\boldsymbol{w}_p$ can be obtained by constructing a *topic space* $\mathcal{T}$ using models such as TF-IDF, Latent Semantic Analysis (LSA) [132], or Latent Dirichlet Allocation (LDA) [12, 44]. Construction of the topic space produces vector representations of the prompt $\boldsymbol{w}_p^{(1:Q)}$. Once the topic space has been constructed (ie: the models have been trained), a vector representation of a test response can be obtained and vector distance metrics used to assess relevance. These models can be trained on either the text of the prompts $\boldsymbol{w}_p^{1:Q}$ or the transcriptions of example responses to these prompts, yielding either a prompt-based or response-based topic space.

In the latter case, multiple example responses to a particular prompt are merged into one 'aggregate' example response to increase the robustness of the vector representation of the prompt. The advantage of a purely response-based representation of prompts is that there are typically many responses to a prompt, which means that there is far more text to train the model on, allowing the topic space to be more robustly estimated. Furthermore, the structure of *spoken* language is quite different from *written language*. It is perfectly normal to have pauses, repetitions and re-starts in spoken language, but not necessarily in written language. As a consequence, there will be a certain mismatch between written prompts and transcriptions of *spoken* responses. Defining the vector representations exclusively using responses would eliminate that mismatch.

While robust vector representations of prompts can be derived from aggregated example responses, the *diversity* of possible responses to a prompt is lost. One approach to capturing the diversity of responses would be to project each example response using a trained model. This retains the robust definition of the topic space while allowing each prompt to be represented by a cloud of points, thereby capturing the diversity of possible responses to a prompt. A K-nearest Neighbor (KNN) classifier can be used to classify the prompt to which a response belongs by computing distances of the test response to each of the training points in the space. The KNN classifier can be modified to yield the K-best[2] classification by removing all training points from the 1-best class from the KNN classifier and re-running the classification to get the 2-best results, and so on.

Approaches based on these principles were first applied to spoken assessment in [131] and then in [31]. The detection of responses for which an automatic assessment system will have difficulty in assigning a valid score, of which non-relevant responses are a specific type, was investigated in [133]. Here, a decision tree classifier was used with features based on cosine similarity between a test response and *tf-idf* representation of both aggregate example responses and questions, as well as pronunciation and fluency features. In [30] detection of text reuse and plagiarism using a decision tree classifier based on vector similarity and lexical matching features, which compare a response to a set of example 'source texts', was investigated. These tasks are similar to prompt-response response relevance assessment in that a classifier is constructed based on vector distance between representations of a test response and either a representation of example (training) responses or the question prompt.

[2]Here there is a conflation of K-nearest neighbours and top-K results. These are different K-s.

8.2.2 Prompt-topic adapted RNN Language Model

Approaches based on distances between vector representation of responses and prompts have previously been successfully applied to a range of tasks, as previously stated. However, there are several deficiencies with this class of approaches. Firstly, they rely on on bag-of-words vector representations, such as TF-IDF, LSA and LDA, which lose sequential information important to evaluating the semantic content of responses. Secondly, while prompt lengths are typically similar, the length of responses can greatly vary - if any of the test responses are short, then their vector representations may not be robustly estimated. Thirdly, it is necessary to select a particular distance metric to assess the distance between the prompt and response representations in topic space. Finally, classification using non-parametric classifiers, such as KNN, becomes impractical for very large datasets, as the number of response vectors scales with the size of the training data. Taking account of response diversity exacerbates this issue.

To address these issues a prompt-topic classification framework based on prompt-topic adapted language models is proposed. A prompt-conditional language model $\mathtt{P}(\boldsymbol{w}_r|y_p = q)$, which is a generative model of the response sentence given the prompt sentence, is used to derive a probability of the prompt via Bayes' rule.

$$\begin{aligned}\mathtt{P}(y_p = q|\boldsymbol{w}_r) &= \frac{\mathtt{P}(\boldsymbol{w}_r|y_p = q)\mathtt{P}(y_p = q)}{\sum_{j=1}^{Q}\mathtt{P}(\boldsymbol{w}_r|y = j)\mathtt{P}(y_p = j)} \\ &\approx \frac{\mathtt{P}(\boldsymbol{w}_r|y_p = q)}{\sum_{j=1}^{Q}\mathtt{P}(\boldsymbol{w}_r|y_p = j)} \\ &\propto \mathtt{P}(\boldsymbol{w}_r|y_p = q)\end{aligned} \tag{8.11}$$

Here, the prior probability of all prompts $\mathtt{P}(y_p = q)$ is assumed to be uniform. Topic classification is done by selecting the prompt-topic which yields the highest probability of generating the given response:

$$\hat{y}_p = \arg\max_q \{\mathtt{P}(\boldsymbol{w}_r|y_p = q)\} \tag{8.12}$$

Given the predicted prompt-topic, a response is considered relevant to the prompt if the predicted prompt matches the prompt which was asked on the exam. The accuracy topic classifier can be improved by considering whether the true prompt topic is among the top K predictions. However, this will lead to a higher rate of false positives.

In practice, the language model $\mathtt{P}(\boldsymbol{w}_r|y_p)$ can be constructed using a conditional Recurrent Neural Network language model (RNNLM) which is trained to associate the example responses with points $\boldsymbol{h}_p$ in the topic-space. In practice, a vector representation of the prompt

$\boldsymbol{h}_p$ is used for adaptation of the language model:

$$\mathtt{P}(\boldsymbol{w}_r|y_p = q) \approx \mathtt{P}(\boldsymbol{w}_r|\boldsymbol{h}_p^{(q)}) \tag{8.13}$$

These vector representations can be derived in the same way as for the vector-distance based approaches discussed above. The RNNLM breaks down the probability of a sentence into a product of the probabilities of the next word in a sequence given the current word and a history vector, which encodes the full back-history of the sentence.

$$\begin{aligned} \mathtt{P}(\boldsymbol{w}_r|\boldsymbol{h}_p^{(q)};\hat{\boldsymbol{\theta}}) &= \prod_{t=1}^{T} \mathtt{P}(w_r^{(t)}|w_r^{(0:t-1)}, \boldsymbol{h}_p^{(q)};\hat{\boldsymbol{\theta}}) \\ \mathtt{P}(w_r^{(t)}|w_r^{(0:t-1)}, \boldsymbol{h}_p^{(q)};\hat{\boldsymbol{\theta}}) &= f(\boldsymbol{h}^{(t)}, \boldsymbol{h}_p^{(q)};\hat{\boldsymbol{\theta}}) \\ \boldsymbol{h}^{(t)} &= f(w^{(t-1)}, \boldsymbol{h}^{(t-1)}, \boldsymbol{h}_p^{(q)};\hat{\boldsymbol{\theta}}) \end{aligned} \tag{8.14}$$

This conditional language model needs to be trained on a corpus of transcriptions of spoken responses and associated vector representations of the prompts asked $\mathcal{D}_{\mathtt{trn}}$:

$$\mathcal{D}_{\mathtt{trn}} = \left\{\boldsymbol{w}_r^{(i)}, y_p^{(i)}\right\}_{i=1}^{N} = \hat{\mathtt{p}}_{\mathtt{trn}}(\boldsymbol{w}_r, y_p) \tag{8.15}$$

The model is trained by minimizing the expectation with respect to the empirical distribution $\hat{\mathtt{p}}_{trn}(\boldsymbol{w}_r, y_p)$ of the negative log-likelihood of the sentences:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}, \mathcal{D}_{\mathtt{trn}}) &= -\mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{trn}}(\boldsymbol{w}_r, y_p)}\Big[\ln \mathtt{P}(\boldsymbol{w}_r|\boldsymbol{h}_p^{(y_p)})\Big] \\ &= -\mathbb{E}_{\hat{\mathtt{p}}_{\mathtt{trn}}(\boldsymbol{w}_r, y_p)}\Big[\sum_{t=1}^{T} \ln \mathtt{P}(w_r^{(t)}|w_r^{(0:t-1)}, \boldsymbol{h}_p^{(y_p)})\Big] \end{aligned} \tag{8.16}$$

This approach has several benefits over standard vector-distance approaches. Firstly, this approach explicitly takes account of word order of the response in order to predict the most likely to have generated this response. Secondly, there is no need to explicitly select a distance metric. Thirdly, the problems of robustly estimating a vector representation $\boldsymbol{h}^r$ of the test response are sidestepped. Furthermore, the RNNLM accounts for a broad range of responses because it is trained on individual response utterances which it associates with a prompt vector. This makes it more scalable than the KNN approach because the number of comparisons which need to be made scales only with the number of questions, not the size of the training data. Thus, arbitrarily large data sets can be used to train the model without affecting classification time. However, this may still be computationally expensive

if there is a large number of prompts in the exam. A limitation of this approach is that it requires prompt-response pairs for all prompts in the examination and can only assess relevance to prompts which they have seen in the training data. This limits the flexibility and increases the cost of deployment of such systems, as example responses have to be collected for newly introduced prompts. Re-training the system to include new prompts may also be computationally costly.

8.3 Direct Prompt-Response Relevance Assessment

Having discussed indirect approaches to prompt-response relevance assessment, we now consider models which directly yield a relevance score $\mathcal{S}(\boldsymbol{w}_r, \boldsymbol{w}_p)$ between a prompt and a response. The simplest approach to consider would be to use the semantic distance $\mathtt{D}_{\mathtt{sem}}$ between vector representations of prompts and responses, discussed in the previous section, as an (inverse) relevance score:

$$\mathcal{S}(\boldsymbol{w}_r, \boldsymbol{w}_p) = \mathtt{D}_{\mathtt{sem}}(\boldsymbol{w}_r, \boldsymbol{w}_p)^{-1} \tag{8.17}$$

However, as was previously discussed, such approaches are limited, as it is necessary to both choose an appropriate distance metric and to use of bag-of-words embeddings of sequences, which leads to large losses of semantic of information. These embeddings are not constructed such that a standard metric of distance between them yield a 'semantic distance' which is optimal for prompt-response relevance.

Instead of using bag-of-words embeddings and choosing a particular distance metric, it is possible to use neural-network based discriminative *metric learning* approaches, such as *Siamese Neural Networks* [14]. These models learn to map a pair of inputs $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ into a shared representation space where Euclidean distance *is* a meaningful measure of semantic distance. Such models are typically formulated as 2-class probabilistic discriminative models which yield the probability of 'similarity'. For prompt-response relevance assessment a Siamese neural network would map the prompt and response sequences into a shared space to yield the probability of relevance:

$$\mathtt{P}(\mathtt{rel}|\boldsymbol{w}_r, \boldsymbol{w}_p; \hat{\boldsymbol{\theta}}) \tag{8.18}$$

These models are typically trained using a standard negative log-likelihood loss function:

$$\mathcal{L}^{NLL}(\boldsymbol{\theta}, \mathcal{D}_{trn}) = \mathbb{E}_{\hat{\mathtt{p}}(\mathtt{rel}, \boldsymbol{w}_p, \boldsymbol{w}_r)}\big[-\ln(\mathtt{P}(\mathtt{rel}|\boldsymbol{w}_r, \boldsymbol{w}_p; \boldsymbol{\theta})\big] \tag{8.19}$$

Unlike topic-classification models, which simply require prompt-topic labelled prompt-response pairs, these models must be trained on a balanced dataset $\mathcal{D}_{trn}$ containing both positive (relevant) and negative (non-relevant) prompt-response pairs. In the absence of true non-relevant prompt-response pairs, negative examples can be constructed by mixing prompts and responses from different prompt-response pairs.

Previously, such models have been applied to the task of prompt-response relevance assessment [134, 72]. A limitation of these approaches when processing sequence data is that *variable* length sequences are mapped into a space of *fixed* dimensionality. This can results in a degradation in performance when processing very long sequences. However, recent work in the fields of Neural Machine Translation and Question Answering [8, 43] has produced a number of *attention-based* deep learning architectures to overcome this problem. As discussed in section 2.1.3, the key advantage of these models is their ability to use an attention mechanism to optimally extract relevant information from a variable-length sequence into a fixed-length embedding. This allows these models to effectively process both short and long sequences.

In this chapter we consider two discriminative relevance assessment models which combine elements of Siamese-network style and attention-based approaches. Specifically, a model which uses an attention mechanism to extract an optimal embedding of a response sequence itself based on an embedding of the prompt sequence is considered in section 8.3.1. An extension of this model which aims to improve performance on unseen prompts in detailed in section 8.3.2.

8.3.1 Attention-based Discriminative Models

This section describes a discriminative, neural Attention-based Model (ATM) for directly assessing the relevance of responses to prompts. The model is illustrated in figure 8.2. It consists of four components; a prompt encoder, a response encoder, an attention mechanism and a binary classifier. The proposed model directly assesses the relevance of responses to prompts by using the prompt $\boldsymbol{w}_p$ to extract information from the response $\boldsymbol{w}_r$ which is used to assign a probability of relevance. This is accomplished by learning to dynamically compute a representation (embedding) $\boldsymbol{h}_p$ of the prompt using the prompt encoder. This prompt embedding is used to attend over a variable-length representation (embedding) of the response $\boldsymbol{h}_r^{(1:T)}$ via an attention mechanism, which should highlight the parts of the response most relevant to the prompt. Based on this information, a binary classifier assigns the probability of the response being relevant to the prompt. As this is now a discriminative model, it is possible to derive measures of *data uncertainty* in the form of the entropy of the

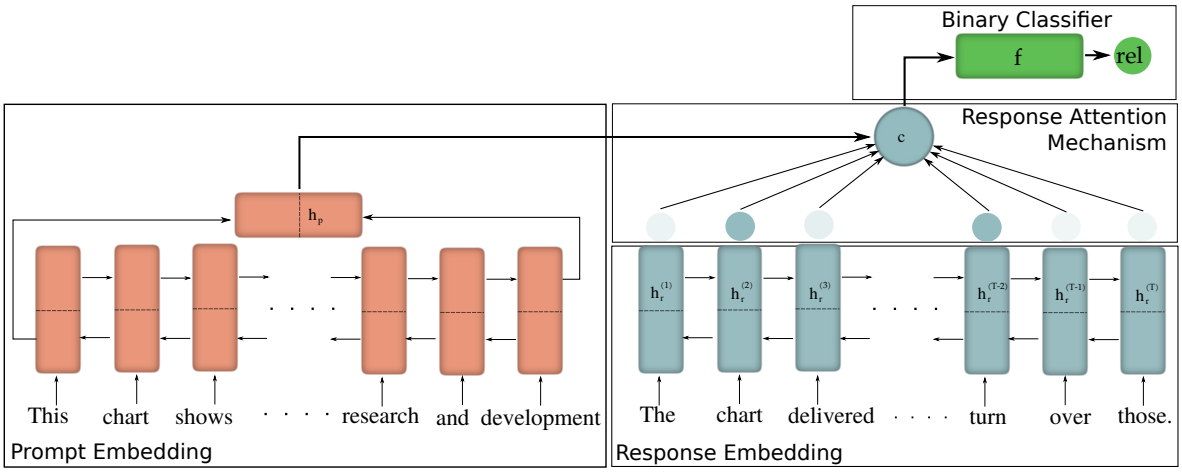


Figure 8.2 Attention-based Direct Relevance Assessment Model

output. Measures of *knowledge uncertainty* can be obtained by considering ensembles of ATM models.

The construction of the model is described here in greater detail. The prompt (eq. 8.20) and response (eq. 8.21) encoders are Bidirectional Recurrent Neural Networks [108] with Long Short-Term Memory (LSTM) recurrent units [57, 42] which process the word sequences of the prompt $\boldsymbol{w}_p = \{w_p^{(1)}, \cdots, w_p^{(L)}\}$ and response $\boldsymbol{w}_r = \{w_r^{(1)}, \cdots, w_r^{(T)}\}$, respectively. As discussed in section 2.1.2, Bi-LSTMs are currently the standard architecture for processing variable-length sequences. Given a Bi-LSTM which has processed the prompt-sequence, a fixed-length embedding of the prompt $\boldsymbol{h}_p$ is computed by concatenating the final forward in time $\overrightarrow{\boldsymbol{h}}_p^{(L)}$ and backward in time $\overleftarrow{\boldsymbol{h}}_p^{(1)}$ hidden states of the prompt encoder.

$$\begin{aligned} \overrightarrow{\boldsymbol{h}}_p^{(1:L)} &= \overrightarrow{\texttt{LSTM}}(\boldsymbol{w}_p;\ \boldsymbol{\theta}_p) \\ \overleftarrow{\boldsymbol{h}}_p^{(1:L)} &= \overleftarrow{\texttt{LSTM}}(\boldsymbol{w}_p;\ \boldsymbol{\theta}_p) \\ \boldsymbol{h}_p &= \begin{bmatrix} \overrightarrow{\boldsymbol{h}}_p^{(L)} \\ \overleftarrow{\boldsymbol{h}}_p^{(1)} \end{bmatrix} \end{aligned} \tag{8.20}$$

The forward in time $\overrightarrow{\boldsymbol{h}}_r^{(t)}$ and backward in time $\overleftarrow{\boldsymbol{h}}_r^{(t)}$ hidden states of the response encoder are concatenated at every time step to produce a hidden state $\boldsymbol{h}_r^{(t)}$, which contains information about how the surrounding context relates to the current word. Both the prompt and response embeddings are sensitive to word order, and so, in the same fashion as the topic-adapted

RNNLM, should be able to take word order and syntax into account.

$$\begin{aligned} \overrightarrow{\boldsymbol{h}}_r^{(1:T)} &= \overrightarrow{\texttt{LSTM}}(\boldsymbol{w}_r;\ \boldsymbol{\theta}_r) \\ \overleftarrow{\boldsymbol{h}}_r^{(1:T)} &= \overleftarrow{\texttt{LSTM}}(\boldsymbol{w}_r;\ \boldsymbol{\theta}_r) \\ \boldsymbol{h}_r^{(t)} &= \begin{bmatrix} \overrightarrow{\boldsymbol{h}}_r^{(t)} \\ \overleftarrow{\boldsymbol{h}}_r^{(t)} \end{bmatrix} \end{aligned} \tag{8.21}$$

A fixed-length prompt-conditional embedding $\boldsymbol{c}$ of the response is computed as a weighted sum of the hidden states $\boldsymbol{h}_r^{(t)}$ of the response encoder given a set of attention weights π_t (eq. 8.22) produced by an attention mechanism. The aim of the attention mechanism is to focus only on the properties of the response which are needed to assess relevance, discarding everything which is not necessary.

$$\boldsymbol{c} = \sum_{t=1}^{T} \pi_t \boldsymbol{h}_r^{(t)},\ \pi_t \geq 0, \sum_{t=1}^{T} \pi_t = 1 \tag{8.22}$$

The attention weights for each hidden state are computed as a softmax (eq. 8.23), where the logits are given by a similarity function between the prompt embedding and the response hidden state.

$$\pi_t = \frac{e^{s(\boldsymbol{h}_p, \boldsymbol{h}_r^{(t)})}}{\sum_{\tau=1}^{T} e^{s(\boldsymbol{h}_p, \boldsymbol{h}_r^{(\tau)})}} \tag{8.23}$$

The similarity function (eq. 8.24) computes how strongly a hidden state of the response encoder relates to the embedding of the prompt.

$$s(\boldsymbol{h}_p, \boldsymbol{h}_r^{(t)}) = \boldsymbol{v}_e^{\mathsf{T}} \tanh(\boldsymbol{\Lambda}_1 \boldsymbol{h}_p + \boldsymbol{\Lambda}_2 \boldsymbol{h}_r^{(t)} + \boldsymbol{b}) \tag{8.24}$$

The parameters of the attention mechanism are $\boldsymbol{\theta}_a = \{\boldsymbol{v}_e, \boldsymbol{\Lambda}_1, \boldsymbol{\Lambda}_2, \boldsymbol{b}\}$. This similarity function was used in [8] for neural machine translation. Alternative attention mechanisms, with different similarity functions [77] and attention sharpening [43] could potentially be used, but are not explored in this work.

The fixed-length response embedding $\boldsymbol{c}$ is the fed into a binary classifier f (eq. 8.25) which outputs the probability $\texttt{P}(\texttt{rel}|\boldsymbol{w}_r, \boldsymbol{w}_p)$ of the response relating to the question. In this work f is a deep neural network (DNN) with parameters $\boldsymbol{\theta}_f$.

$$\texttt{P}(\texttt{rel}|\boldsymbol{w}_r, \boldsymbol{w}_p; \hat{\boldsymbol{\theta}}) = f(\boldsymbol{c};\ \boldsymbol{\theta}_f) \tag{8.25}$$

The primary advantage of the proposed model is that it directly assesses relevance of responses to prompts and can embed any prompt into the appropriate space via the prompt encoder. This eliminates the need to pre-compute a set of prompt representations from examples responses, as was done for the topic-adapted RNNLM and the vector-similarity approaches discussed in the previous section. Conceptually, the prompt encoder learns to project the prompt sentence into a prompt-based topic space and the response encoder projects the response sentence into a response-based topic space, similarly to the previous approaches. However, the difference is that these are no longer bag-of-words representations and they are learned *explicitly* for the task of relevance assessment. All components of the model are trained *jointly*, which allows them to learn the necessary representations and transformations which make this possible. This should allow the model to also assess the relevance of responses to newly introduced prompts without the need to collect example responses to the new prompt or for the model to be re-trained. However, the model needs to be trained to generalize well in order to effectively handle unseen prompts, especially if they are quite different to the prompts seen in the training data. Finally, this model is computationally more efficient than the prompt-adapted RNNLM or KNN models, as it only needs to be run once, rather than as many times as there are prompts in the training data.

8.3.2 Hierarchical Attention-based Topic Model

In any spoken language proficiency exam some prompts may be similar to others. Furthermore, newly introduced prompts often are simply old prompts phrased slightly differently. If a model is able to identify that a newly introduced prompt is similar to one or more prompts seen in training (known prompts), then the model may be able to leverage that information to do better assessment of relevance between the new prompt and the candidate's response. Theoretically, the ATM model described above is able to directly assess the relevance between *arbitrary* prompt-response pairs because the response encoder *implicitly* takes advantage of the similarity between prompts. However, if the training dataset contains a small number of diverse prompts, then the model may generalize poorly to new and unseen prompts, leading to poor relevance assessment performance. Therefore, it may be useful to have the model *explicitly* exploit the similarity of the input prompt to each of the prompts seen in the training data. This section describes a model, called the Hierarchical Attention-based Model (HATM), which is constructed to do this. The HATM is an extension of the ATM model described above, where a *prompt attention mechanism* is added to the model, as shown in figure 8.3-purple. This prompt attention mechanism expresses the input prompt as a convex combination of the embeddings of prompts seen in the training data. The HATM views known prompts as the vertexes of a simplex within which all valid prompt embeddings must

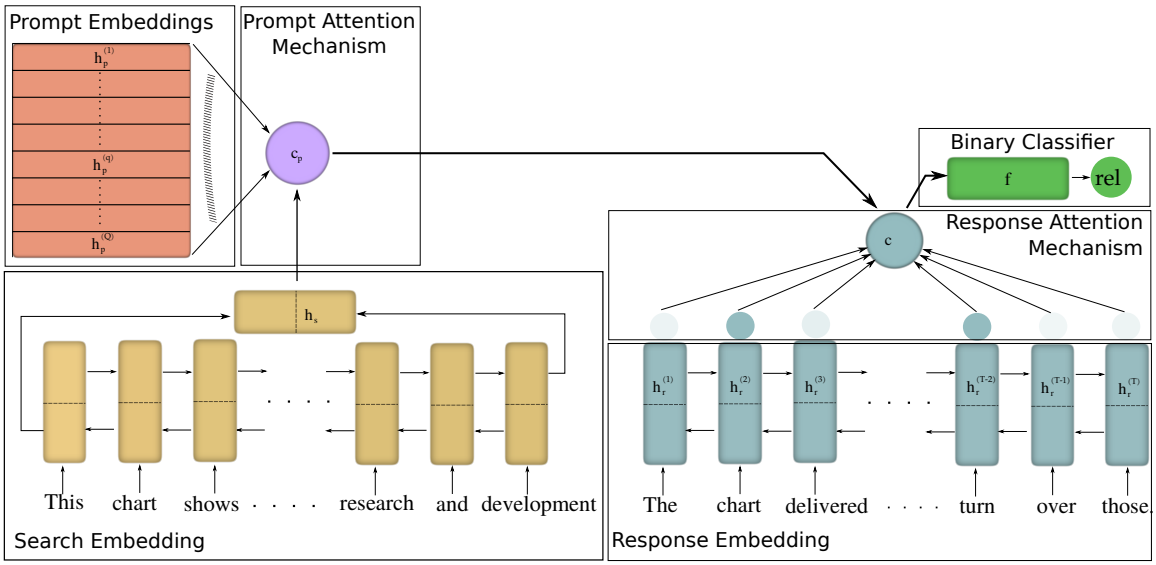


Figure 8.3 Hierarchical Attention-based Direct Relevance Assessment Model

exist. This potentially allows the HATM, given a diverse set of prompt embeddings, to estimate embeddings for unseen prompts more robustly.

This mechanism is now described in greater detail. The prompt attention mechanism attends over the embeddings on prompts seen in the training data produced by an ATM's prompt encoder. The prompt attention mechanism is conditioned on a 'search' embedding $\boldsymbol{h}_s$ of an input prompt produced by new bi-LSTM *search encoder*, shown in figure 8.3-yellow.

$$\begin{aligned} \overrightarrow{\boldsymbol{h}}_s^{(1:L)} &= \overrightarrow{\texttt{LSTM}}(\boldsymbol{w}_p;\ \boldsymbol{\theta}_s) \\ \overleftarrow{\boldsymbol{h}}_s^{(1:L)} &= \overleftarrow{\texttt{LSTM}}(\boldsymbol{w}_p;\ \boldsymbol{\theta}_s) \\ \boldsymbol{h}_s &= \begin{bmatrix} \overrightarrow{\boldsymbol{h}}_s^{(L)} \\ \overleftarrow{\boldsymbol{h}}_s^{(1)} \end{bmatrix} \end{aligned} \tag{8.26}$$

The attention weights π_q are computed via a similarity function $s(\boldsymbol{h}_s, \boldsymbol{h}_p^{(q)})$ between a seen prompt embedding $\boldsymbol{h}_p^{(q)}$ and the 'search' embedding of the input prompt $\boldsymbol{h}_s$:

$$\pi_q = \frac{e^{s(\boldsymbol{h}_s, \boldsymbol{h}_p^{(q)})}}{\sum_{j=1}^{Q} e^{s(\boldsymbol{h}_s, \boldsymbol{h}_p^{(j)})}} \tag{8.27}$$

Given the prompt attention weights, the input prompt $\boldsymbol{w}_p$ is expressed as a convex combination $\boldsymbol{c}^p$ of the embeddings $\boldsymbol{h}_p^{(1:Q)}$:

$$\boldsymbol{c}_p = \sum_{q=1}^{Q} \pi_q \boldsymbol{h}_p^{(q)} \tag{8.28}$$

This prompt embedding is then used to attend over a variable length embedding of the response, as before. The prompt attention mechanism as expressed in equation 8.27 is not necessarily forced to discover the similarities between prompts, as it could learn to yield an identity attention during training. In order to force it to discover the similarities between prompts, the attention mechanism is modified so that *during training* the prompts cannot attend over themselves - the attention mechanism is trained in a 'leave-one-out' fashion to teach it to reconstruct each prompt in the training data from all other seen prompts:

$$\pi_q = \begin{cases} \dfrac{e^{s(\boldsymbol{h}_s, \boldsymbol{h}_p^{(q)})}}{\sum_{q=1, \boldsymbol{w}_p \neq \boldsymbol{w}_p^{(q)}}^{Q} e^{s(\boldsymbol{h}_s, \boldsymbol{h}_p^{(q)})}}, & \text{if } \boldsymbol{w}_p \neq \boldsymbol{w}_p^{(q)} \\ 0, & \text{if } \boldsymbol{w}_p = \boldsymbol{w}_p^{(q)} \end{cases} \tag{8.29}$$

However, no masking is used during inference.

An interesting property of the HATM is that it is possible to derive measures of uncertainty from the prompt attention mechanism, as the dimensionality (number of seen prompts) is fixed. Firstly, the entropy of the prompt attention mechanism can be used as a measure of one aspect of *data uncertainty* in the prompt embedding:

$$\mathcal{H}[\boldsymbol{\pi}|\boldsymbol{w}_p] = -\sum_{q=1}^{Q} \pi_q \ln \pi_q \tag{8.30}$$

If the entropy of the attention mechanism is high, then it indicates that the input prompt is equally similar (or dissimilar) to all of the prompts seen in training data. On the other hand, if the entropy of the attention mechanism is low, then the model has chosen to focus on a few of the most similar prompts to the prompt in question.

Secondly, given an ensemble of HATM models $\left\{\mathtt{P}(\mathtt{rel}|\boldsymbol{w}_r, \boldsymbol{w}_p; \boldsymbol{\theta}^{(m)})\right\}_{m=1}^{M}$ it is possible to calculate the mutual information, a measure of *knowledge uncertainty*. The mutual information can be expressed as the difference of the entropy of the expected prompt attention, a measure of *total uncertainty*, and expected entropy of the prompt attention

weights, a measure of *data uncertainty*:

$$\mathcal{I}[\boldsymbol{\pi}, \mathcal{M}|\boldsymbol{w}_p] \approx \mathcal{H}\Big[\frac{1}{M}\sum_{m=1}^{M}\boldsymbol{\pi}^{(m)}\Big] - \frac{1}{M}\sum_{m=1}^{M}\mathcal{H}[\boldsymbol{\pi}^{(m)}] \tag{8.31}$$

As a measure of *knowledge uncertainty*, mutual information assesses whether the model understands the input prompt. At the same time, the entropy of the expected attention weights is a measure of *total uncertainty* in the prompt embedding. These measures of uncertainty can be used to assess whether the HATM is understands the input prompt. In addition, it is also possible to calculate the expected pairwise KL-divergence between the prompt attention weights of all models in the ensembles

$$\mathcal{K}[\boldsymbol{\pi}, \mathcal{M}|\boldsymbol{w}_p] \approx \frac{1}{M^2}\sum_{m=1}^{M}\sum_{j=1}^{M}\mathtt{KL}[\boldsymbol{\pi}^{(m)}||\boldsymbol{\pi}^{(j)}] \tag{8.32}$$

These approaches are similar to deriving measures of uncertainty from the output of a model. However, the difference is that while measure of uncertainty derived from the output give the overall uncertainty in the *prediction*, measures of uncertainty derived from the attention mechanism yield uncertainty in the *prompt representation*. The advantage of deriving estimates of uncertainty in the prompt representation is that they allow the designer of the system to know how well the model will perform when assessing relevance to a particular prompt without having to gather real responses to the new prompt. This affords the model designer a significant operational advantage, as it is possible to identify the new prompts for which the designer will need to collect training responses before before deploying the system.

8.4 Experiments: Indirect Relevance Assessment

The current section evaluates indirect approaches to prompt-response relevance assessment, discussed in section 8.2, on spoken responses to question-prompts from the BULATS exam [15]. Two forms of experiment are conducted in order to assess the performance of the models. Firstly, a prompt classification experiment is run, where the ability of the system to accurately recognize the topic of a response is evaluated. Secondly, a prompt-response relevance assessment experiment, which considers both false-positives and false negatives, is conducted.

Note, that the work described in this section is the earliest done as part of this thesis, and therefore is not fully comparable to the experiments on direct approaches considered

in section 8.5, due to both a data mismatch, and 'less advanced' deep learning approaches being considered.

8.4.1 Description of training and evaluation datasets

In this section data from the Business Language Testing Service (BULATS) English tests is used for training and testing of indirect relevance assessment models. This work focuses on the 3 sections of the BULATS exam where open ended prompts elicit spontaneously constructed responses. In Section C, candidates talk about a work related topic; in section D candidates must describe a graph such as a pie or bar chart related to a business situation; in section E candidates are asked to respond to 5 prompts related to a single context prompt. The section E sub-prompts are sufficiently distinct to merit their own prompt vector representations. However, at classification time confusions between sub-prompts of an overall section E prompt are not considered mistakes.

In the following experiments two training datasets TRN1 and TRN2 are considered. TRN1 consists of ASR transcriptions 9.9K responses from 490 candidates, with an average 35.1 responses per prompt. TRN2 is a larger dataset which contains 202K responses from 10004 candidates, with an average 715.5 responses per prompt. Both datasets contain responses to 282 unique prompts from *all* 5 sections of the BULATS exam. Table 8.1 shows the number of unique prompts per section of the BULATS exam contained in TRN1 and TRN2.

Section	A	B	C	D	E
Uniq. Prompts	18	144	17	18	85
Words/Resp.	10	10	61	77	20

Table 8.1 Data Characteristics

As table 8.1 shows, the average response length varies across sections due to the nature of the sections. Shorter responses are observed for sections A, B and E, with longer responses to C and D. Estimating topic representations for sections A, B and E questions based on individual responses would be problematic due to the short response lengths. Further details of the datasets are available in table 8.2. In this section two held-out data sets are used for development, *DEV*, and evaluation, *EVL*, composed of 572 and 1502 responses to section C, D and E prompts. All datasets are transcribed using an ASR system, detailed in [82], which has a word error rate (% WER) on 31.5 on the DEV dataset relative to a set of professionally produced transcriptions, *DEV REF*. The version of DEV with professional transcriptions is

Data	#Prompt	#Resp.	#Words	#Resp./ Prompt	Avg.Resp. Length
TRN1	282	9.9K	200.6K	35.1	20.3
TRN2		202K	4.1M	715.5	20.3
DEV	92	572	31K	6.2	54.2
EVL		1502	71.9K	14.1	47.9

Table 8.2 Prompt, response and word statistics of the prompt-response BULATS datasets based on 1-best recognition hypotheses.

also used to assess the effect of ASR error rates on prompt-classification performance. All datasets considered here are composed of predominantly Gujarati L1 candidates.

8.4.2 Model Construction

In the current set of experiments all models use 280-dimensional LSA [69, 92] embeddings of prompts derived from concatenated example responses from TRN1. The scikit-learn version 17.0 toolkit [100] implementation of LSA was used. As shown in table 8.2, there are on average 715.5 words per prompt, when responses are concatenated, but only an average of 20-80 words per response, as shown in table8.1. Thus, using concatenated responses should yield a more robust response-based prompt embedding.

Three models were considered. Firstly, a KNN-classifier was constructed based on cosine distances between LSA embeddings of prompts. Here, *individual* example responses to each prompt in TRN1 were projected into the LSA-based vector space which was already trained on concatenated example responses. This was done so that there would be a cloud of points representing each prompt, thereby capturing the diversity of possible responses, as discussed in section 8.2.1. The scikit-learn implementation of an KNN classifier with distance-weighted voting was used. The KNN model was only trained on the TRN1 dataset, as inference was far too slow using the much larger TRN2 dataset.

Secondly, two prompt-adapted RNNLM models RNN1 and RNN2 are trained. RNN1 is trained on TRN1 and RNN2 is trained on TRN2. RNN1 uses a 100-dimensional hidden layer while RNN2 uses a 512 dimensional hidden later. Both models had an input-vocabulary size of 62.4K and an output vocabulary size of 46.8K unique words. Both models are trained using the back-propagation through time (BPTT) algorithm [87, 88, 86], using stochastic gradient descent with a minibatch size of 64 and an initial learning rate of 1.0. DEV REF is used as a validation data set for early stopping to prevent over-fitting. The prompt-adapted

RNNLM systems were implemented using the CUED RNNLM toolkit v0.1 [19], details of which can be found in [20, 87].

8.4.3 Prompt Classification

In the current section the prompt-topic classification performance of the KNN, RNN1 and RNN2 systems is assessed on ASR and professional transcriptions of the development DEV dataset. The results are quoted in terms of the classification error rate, which can also be interpreted as the false-rejection rate. As previously stated, in the current set of experiments confusions between sub-prompts of the same overall section E prompt are *not* considered a misclassification.

The results are presented in table 8.3, where several trends can be observed. Firstly, the RNN1 system outperforms the KNN system only marginally, mostly due to better performance on section E. At the same time, the RNN2 system, which is trained on 20 times more data, yields greatly improved performance over KNN and RNN1 systems, almost halving the overall error rate. The KNN system could not be evaluated effectively in reasonable time using 20 times as many example responses and results are not shown, while RNN2 evaluation times are unaffected. This illustrates how non-parametric KNN systems, while simple and easy to implement, suffer scaling issues. Secondly, there is only a minor improvement in terms of classification performance by using professional transcriptions of DEV. This suggests that the systems are quite robust to a high degree of ASR error, but at the same time are unable to make better use of higher quality transcriptions. Thirdly, the performance of all models is always best on section D. This is likely due to section D prompts being very distinct, as they are about describing charts and graphs. At the same time section C and E questions are the less distinct because they have free-form answers to broad questions, leading to higher response variability. This makes the linking of the prompt from the training data to the test data more challenging, particularly for 1-best classification, leading to higher error rates. Notably, the RNN1 and RNN2 systems perform better on section E by about 5-10% than on section C. Clearly, this suggests that section C questions are hardest to assess.

The typical mistakes which the models make can be analyzed by means of a confusion matrix. Figure 8.4 shows which *sections* are most often confused by the RNN1 system. The confusion matrix for the KNN and RNN2 systems are similar. Figure 8.4 shows that most confusions are with prompts in same sections. Section C and D prompts are almost never confused with prompts from other sections, while section E prompts are sometimes confused with section C prompts about %12 percent of the time. This is likely because each section has a distinct style of prompts and some prompts within a section are similar. An example is

Topic Repn.	System	# Cands.	C		D		E		ALL (C-E)	
			REF	ASR	REF	ASR	REF	ASR	REF	ASR
LSA	KNN	490	32.1	28.6	2.5	3.7	31.3	33.3	22.0	21.9
	RNN1		29.8	31.0	4.9	6.2	23.8	23.8	19.7	20.5
	RNN2	10004	19.0	19.0	3.7	3.7	9.5	10.7	10.8	11.2

Table 8.3 % False rejection rate in prompt classification on the DEV dataset. KNN classifier uses 6 nearest neighbour and distance weighting.

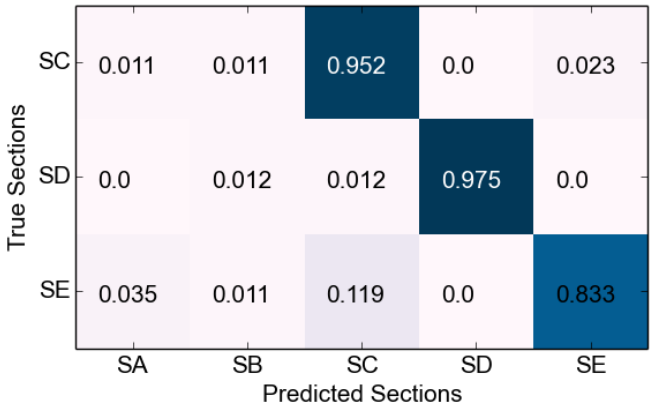


Figure 8.4 Section confusion matrix of RNN1 system on DEV ASR.

shown below. Prompt SC-EX1 relates to personal local events in the workplace. SC-EX2, which relates to similar issues, is often confused with it. On the other hand, SC-EX3 is rarely confused with SC-EX1 as it is about non-personal events on a larger scale.

- SC-EX1: Talk about some advice from a colleague. You should say: what the advice was, how it helped you and whether you would give the same advice to another colleague.
- SC-EX2: Talk about a successful day you had at work. You should say: what work you did on that day, how you dealt with any difficulties and why the day was successful.
- SC-EX3: Talk about a company which is a major employer in your town or a town near you. You should say: what the business of the company is, what kind of jobs it offers and how the local community feels about the company.

8.4.4 Prompt-Response Relevance Assessment

In this section prompt-response relevance assessment is investigated by considering the false-rejection (FR) of matched prompt-response pairs and *false-acceptance* (FA) rates of mismatched prompt-response pairs. Here, instead of considering the 1-best prediction of the prompt classifier, the false-acceptance and false-rejection rates are evaluated for K-best predictions as the value of K is increased. This experiment is conducted on the EVL dataset.

Since the datasets considered here do not contain real non-relevant responses, a pool $\boldsymbol{W}_r^{(q)}$ of *mismatched* responses is synthetically generated for each prompt by using valid responses to other prompts. Non-relevant responses are then selected from this pool. A selection strategy defines which responses are present in $\boldsymbol{W}_r^{(q)}$. Rather than using a single selection of non-relevant responses, an expected performance over all possible non-relevant response selections is estimated. The overall probability of falsely accepting a non-relevant response can be expressed as follows:

$$\begin{aligned} \mathtt{P}(\mathtt{FA}) &= \sum_{q=1}^{Q} \sum_{\boldsymbol{w}_r \in \boldsymbol{W}_r^{(q)}} \mathtt{P}(\mathtt{FA}|\boldsymbol{w}_r, q)\mathtt{P}(\boldsymbol{w}_r|y_p = q)\mathtt{P}(q) \\ &= \frac{1}{Q}\sum_{q=1}^{Q} \frac{1}{|\boldsymbol{W}_r^{(q)}|} \sum_{\boldsymbol{w}_r \in \boldsymbol{W}_r^{(q)}} \mathtt{P}(\mathtt{FA}|\boldsymbol{w}_r, y_p = q) \end{aligned} \tag{8.33}$$

In equation 8.33, a prompt q is selected with uniform probability from the set Q of possible prompts. Then a response $\boldsymbol{w}_r$ is randomly selected with uniform probability $\mathtt{P}(\boldsymbol{w}_r|q)$ from the pool of mismatched responses to the prompt $\boldsymbol{W}_r^{(q)}$. The correct responses to the prompt are *not* present in the pool. The conditional probability of false accept $\mathtt{P}(\mathtt{FA}|\boldsymbol{w}_r, q)$ is defined as:

$$\mathtt{P}(\mathtt{FA}|\boldsymbol{w}_r, y_p = q) = \begin{cases} 1, \ y_p \in \{\hat{y}_1, \cdots, \hat{y}_K\} \\ 0, \ y_p \notin \{\hat{y}_1, \cdots, \hat{y}_K\} \end{cases} \tag{8.34}$$

As shown in figure 8.4, the main confusions will occur between prompts within the same section. Two strategies for selecting non-relevant responses are considered based on this: *naive*, where a mismatched response can be selected from any section; and *directed*, where a mismatched response can only be selected from the same section as the prompt. The *naive* strategy more closely represents either candidates who simply have made a mistake and misunderstood the prompt or malicious candidates who have memorized an unrelated English passage from a book or newspaper for example. On the other hand, the *directed* strategy represents malicious candidates who are familiar with the test system, have access to real

responses from previous tests and are actively seeking to cheat on their spoken proficiency exams.

The false acceptance (FA) and false rejection (FR) rates for range of K-best classification results are plotted on a curve in figure 8.5, which shows several notable trends. Firstly, it shows that the *directed* strategy clearly yields higher false acceptance rates for the same false-rejection rate as the *naive* strategy. Secondly, the RNN1 and RNN2 systems consistently outperform the KNN system at all operating points and for both selection strategies. Furthermore, the RNN2 system significantly outperforms the RNN1 system, which shows how using the deep-learning based prompt-adapted RNNLM system is able to scale performance using more data, while keeping inference computational cost constant. Finally, the difference between the *naive* and *directed* prompt-shuffling stategy is smallest at all operating points of the RNN2 system, which shows that it is able to discriminate well between similar prompts.

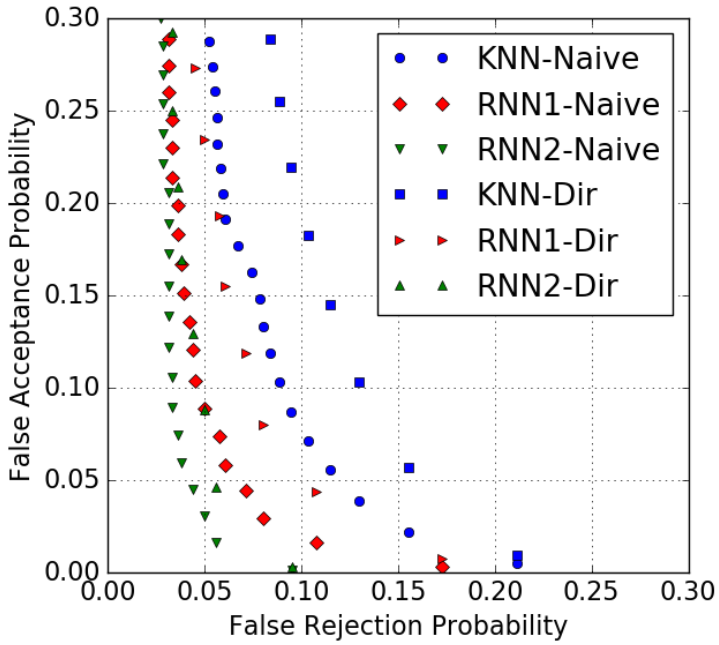


Figure 8.5 False Acceptance vs. False rejection curve on EVL dataset using ASR transcriptions.

The Equal Error Rates (EER), where FA = FR, are given in table 8.4. As expected, the equal error rate for the *directed* strategy is higher than for the *naive* strategy. It is interesting that for better systems the difference in performance against the *naive* and *directed* strategies decreases. This indicates that the systems become increasingly better at discriminating between similar prompts.

In relation to the stated task of assessing prompt-response relevance, the *directed* strategy represents a lower-bound on realistic system performance, as candidates are not likely to

System	% Equal Error Rate	
	Directed	Naive
KNN	12.5	9.0
RNN1	8.0	6.0
RNN2	5.0	4.5

Table 8.4 Equal error rate (EER) operating points where FA = FR on EVL dataset using ASR transcriptions.

respond with a valid response to a different prompt. Most likely they will fail to construct a valid response or will add completely unrelated phrases memorized beforehand, which, unlike responses from other sections, may not come from the same domain as the test (eg: Business for BULATs). However, as discussed in section 8.1, prompt-topic classification may be an excessively harsh approach to prompt-response relevance assessment. Thus, this concludes the discussion of indirect approaches to prompt-response relevance assessment. The next section will consider direct approaches to prompt-response relevance assessment.

8.5 Experiments: Direct Relevance Assessment

The previous section evaluated methods of indirect assessment of prompt-response relevance via prompt-topic classification. It was shown how approaches based on RNN language models are able to outperform a K-nearest neighbour classifier based on vector distance features. Crucially, it was shown how deep learning methods are able to scale to larger datasets and obtain improved performance at no additional test-time computational cost. However, the prompt-adapted RNNLM still needs to be evaluated as many times are there are unique prompts in the training dataset. Thus, in this section the assessment of prompt-response relevance using the *direct* approach, discussed in section 8.3, is investigated. Here, the Attention-based Model (ATM) and Hierarchical Attention-based Model (HATM) are evaluated on data from both the BULATS and LinguaSkill spoken English proficiency exams. As in the previous set of experiments, the text for each response was generated using an Automatic Speech Recognition (ASR) system. The 1-best recognition hypothesis was then passed to a relevance assessment system (ATM/HATM) which assigns a probability of whether the response was relevant to the prompt.

In this section, properties of the data from BULATS and LinguaSkill exams is discussed in section 8.5.1, details of model training are described in section 8.5.3. Investigation of the general properties of the ATM and HATM is done in section 8.5.5, models are evaluated on

their ability to generalize to new and unseen prompts in section 8.5.6 and use of measures of uncertainty derived from an ensemble of HATMs to detect prompt-response pairs which the systems cannot assess well is investigated in section 8.5.7. The experiments described in this section are based on work done in [3, 83].

It is necessary to point out that the direct relevance assessment approaches are not comparable to the in-direct approaches discussed in sections 8.2 and evaluated in section 8.4 for several reasons. Firstly, the approaches operate in different ways and it is difficult to make direct comparison. Secondly, the indirect approaches are the earliest work which is presented in this thesis, which affected the available datasets as well as models. The direct relevance assessment models considered in this section use more advanced deep learning architectures, regularization and training approaches. Furthermore, it is not clear how derive measures of uncertainty for indirect approaches.

8.5.1 Description of training and evaluation datasets

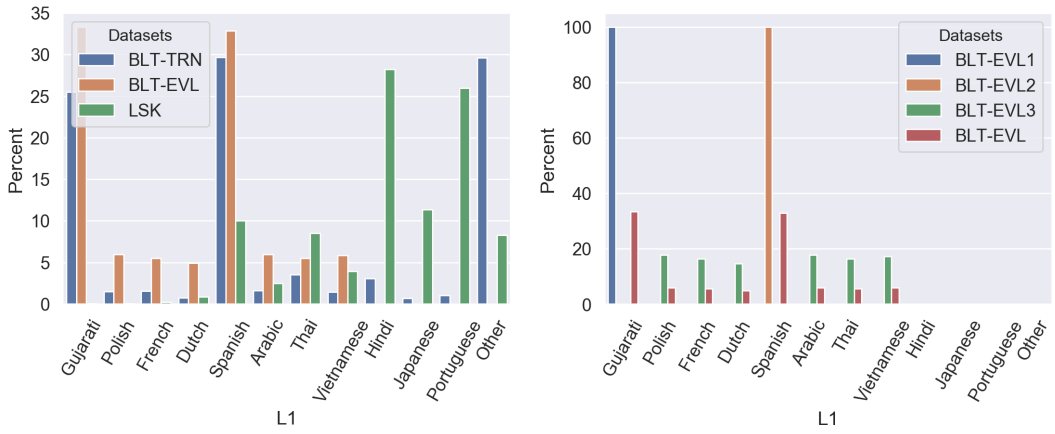
In this section experiments were run on data from both the BULATS and the LinguaSkill speaking exam, described in sections 6.1. As described in section 6.1, the BULATS and LinguaSkill exams have similar structure, but the general domain of the prompts is different. BULATS focuses on business related prompts while LinguaSkill covers on a broad range of subjects. As a results, there is a domain mismatch between the LinguaSkill and BULATS data. This work focuses on the 3 sections where open ended prompts elicit spontaneously constructed responses. In Section C, candidates talk about a work related topic (e.g. the perfect office). Candidates must describe a graph such as a pie or bar chart related to a business situation (e.g. company sales) in Section D. In Section E candidates are asked to respond to 5 prompts related to a single context prompt (e.g. a set of 5 prompts about organizing a stall at a trade fair). There are 7 prompts in total.

Table 8.5 gives the statistics of the prompt-response BULATS and LinguaSkill datasets considered in this section. The training data set *BLT-TRN* contains 13.9M words in 293.0K responses from 42K candidates. 379 unique prompts are seen in *BLT-TRN*, with an average of 773 example responses per prompt and an average response length of 45.8 words. Two evaluating datasets are considered - BLT-EVL, which is derived from prompts and responses to the BULATS exam, and LSK, which is derived from prompts and responses to the LinguaSkill exam. The datasets BLT-EVLS1-3 are subsets of *BLT-EVL*. The prompts in BLT-EVL are a subset of prompts in BLT-TRN, while the prompts in LSK are an entirely different set.

The breakdown of each dataset by the L1 (native) language of the candidates is presented in figure 8.6. Figure 8.6a shows that the training dataset BLT-TRN predominantly contains

Data	#Prompts	#Resp.	#Words	#Resp./ Prompt	Avg.Resp. Length
BLT-TRN	379	293.4K	13.9M	774.6	47.2
BLT-EVLS1	92	1297	64.4K	14.1	49.7
BLT-EVLS2	177	1335	58.5K	7.5	43.8
BLT-EVLS3	179	1445	63.1K	8.1	43.7
BLT-EVL	219	4077	184.5K	18.6	45.3
LSK	56	21.3K	975.0 K	379.8	45.8

Table 8.5 Prompt, response and word statistics of the prompt-response BULATS and LinguaSkill datasets based on 1-best recognition hypotheses using SYS-2 ASR systems.



(a) BULATS training datasets

(b) BULATS evaluation dataset

Figure 8.6 Candidate L1 language distribution of datasets.

Spanish and Gujarati candidates, in addition to candidates with Polish, French, Dutch, Arabic, Thai, Vietnamese and roughly 80 other L1 languages. The BULATS evaluation datasets BLT-EVL is even distributed across Gujarati, Spanish, Polish, French, Dutch, Arabic, Vietnamese and Thai. The LinguaSkill dataset LSK covers a range of L1 languages, notably Dutch, Spanish, Arabic, Vietnamese and Thai, like the BULATS datasets, as well as Japanese, Portuguese, Hindi and roughly 40 other L1 languages. The L1 breakdown of subsets of the BLT-EVL dataset is presented in figure 8.6b, which shows that BLT-EVLS1 only consists of Gujarati speakers and BLT-EVLS2 only of Spanish speakers, while BLT-EVLS3 is uniformly distributed over Polish, French, Arabic, Dutch, Thai and Vietnamese candidates.

The CEFR grade level breakdown of all datasets is given in figure 8.7. Clearly, both the BULATS training dataset and LinguaSkill dataset LSK have an uneven distribution over

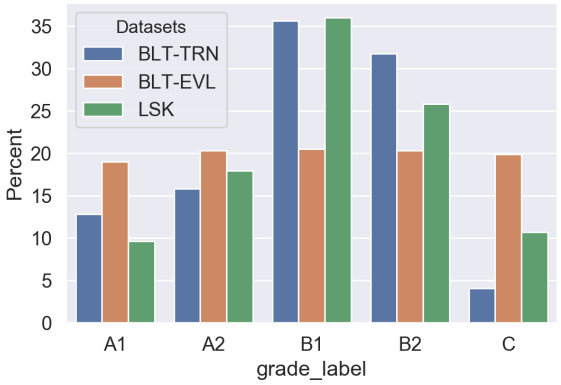


Figure 8.7 CEFR Grade level distribution of datasets

grade levels, with the majority of speakers having B1 and B2 labels. The BULATS evaluation dataset *BLT-EVL* (and its subsets) are uniformly distributed over CEFR grade level.

Two ASR systems, SYS-1 and SYS-2 [3], were used in this section in order to investigate the effect of ASR errors on assessment performance. This is important, as in a real deployment scenario responses of non-native English speakers are transcribed using an ASR system and 1-best hypotheses are used as the text of the response. Both systems are used to transcribe both the training and evaluation data. SYS-1 is a worse ASR system which was only trained using crowd-sourced transcriptions [120] of non-native English speakers whose native language (L1 language) is Gujarati. SYS-2 is trained on crowd-sourced transcriptions of non-native speakers from a range of different L1 languages. The performance of both systems is described in tables 8.6 and 8.7 relative to the crowd-sourced transcriptions [120]. Clearly,

ASR	BLT-EVLS1	BLT-EVLS2	BLT-EVLS3	BLT-EVL	LSK
SYS-1	37.3	52.5	48.6	45.7	-
SYS-2	30.1	30.8	30.4	30.4	37.6

Table 8.6 ASR %WER on evaluation data sets

SYS-2 gives a significant drop in ASR WER% compared in SYS-1, especially on datasets with responses from non-Gujarati L1 candidates, such as BLT-EVLS2, BLT-EVLS3 and LSK-E. SYS-2 has roughly equal performance across all evaluation sets, unlike SYS-1, which is heavily biased towards Gujarati-L1 speakers. Table 8.7 shows that both ASR systems do worse on candidates with a low CEFR proficiency score (A1 and A2). The difference in the performance of both systems is larger on the best speakers than on the worst speakers.

[3]Note, that the ASR system used in indirect relevance assessment experiments was entirely different to the two considered here.

ASR	A1	A2	B1	B2	C	All
SYS-1	60.3	54.0	44.9	41.8	41.4	45.7
SYS-2	49.9	39.4	29.5	27.0	24.4	30.4

Table 8.7 ASR %WER per CEFR grade level on *BLT-EVL*

8.5.2 Training and Evaluation Data Construction

As the data are taken from tests run with human examiners, the responses in all datasets are assumed to be relevant. However, unlike the indirect relevance assessment approaches, which only requires matched prompt-response pairs, direct approaches additionally require negative, non-relevant training examples. Here, these mismatched pairings were produced by shuffling the responses and prompts during training via a dynamic sampling mechanism.

If more than one negative example is shown for a particular response, the positive example is over-sampled the corresponding number of times to maintain a balanced training. For multi-part prompts from section 5 of the BULATS and LinguaSkill exams, which contain a main prompt that describes the overall prompt and several five sub-prompts, all sub-prompts were pre-appended with the main prompt. These sub-prompts are considered distinct topics and thus competing negative examples to each other during shuffling.

Negative examples are also introduced into the evaluation datasets via shuffling. The negative examples are drawn from the empirical prompt distribution of the evaluation data. 10 different shufflings of data are used to construct a diverse set of negative examples for every prompt. The positive examples are simply oversampled 10 times to maintain a balanced set of positive and negatives.

As was shown section 8.4, prompts from the same section tend to be more similar, and therefore more confusable. The same two prompt shuffling strategies considered in section 8.4 are considered in this section. Specifically the *naive* strategy, where prompts are shuffled across all sections of the exam; and the *directed* strategy, where prompts are shuffled only within the same section. *Naive* shuffling corresponds to a more realistic scenario, while *directed* reflects the scenario where the candidate is aware of the relevance detection system and is actively trying to bypass it. For all BULATS evaluation datasets both *naive* and *directed* shuffling is investigated. For the LinguaSkill datasets only naive shuffling is used.

8.5.3 Model and Training Hyper-parameters

The ATM and HATM were implemented in Tensorflow [4] and contain two 400 dimensional BiLSTM encoders with TanH non-linearities, 200 for the forward states and 200 for the

backward states. The HATM also contains an additional 200-dimensional BiLSTM prompt-search encoder. The ATM was trained for 6 epochs with the Adam optimizer [62], an exponentially decaying learning rate with an initial value of 1e-3 and decay factor 0.85 per epoch. Dropout regularization [117] was applied to all layers except for the LSTM recurrent connections and word embeddings, with a keep probability of 0.8. The binary classifier was a DNN with 2 hidden layers of 200 Leaky ReLU units and a 1-dimensional logistic output. The HATM was initialized from a trained ATM. For the first 2 epochs only the newly-initialized prompt-attention mechanism was trained. Further training for 1 more epoch is done with an unlocked response attention mechanism and a learning rate of 1e-4. The prompt and response encoders, as well as the DNN classifier remain locked.

8.5.4 Assessment Criteria

Direct approaches to relevance assessment can be assessed as a binary threshold-based classification task. As was discussed in chapter 5, section 5.2, these tasks can be assessed using area under an ROC curve (AUROC) or a Precision-Recall cure (AUPR). The latter is better for assessment unbalanced datasets. However, in this set of experiments the datasets are balanced (by over-sampling) to have equal numbers of positive and negative examples, so AUROC is used as the metric of performance. Thus, predictive performance of direct relevance assessment approaches will be assessed using AUROC.

In section 8.5.7 derivation of measures of uncertainty is considered. Measures of uncertainty are assessed on the task of misclassification detection at a threshold of 0.5 and via rejection curves. As discussed in chapter 5, misclassification detection can be assessed using area under a Precision-Recall curve, are there are usually fewer misclassifications than correct classification. Rejection curves will be assessed via rejection ratios, where the y-axis corresponds to predictive performance assessed using AUROC.

8.5.5 Performance on Matched Data

The current section investigates the performance of the ATM on evaluation datasets from the BULATS exam. Specifically, the effects of L1 language, ASR system, prompt similarity and CEFR grade-level on relevance assessment performance are assessed. An ensemble of 10 ATM and HATM models was constructed using different random initializations, and the results quoted in this section correspond to the average performance across the ensemble, rather than joint-ensemble performance. Results only for the ATM model are presented, as the behaviour of the HATM model is nearly identical.

Table 8.8 show the AUROC scores for ATM models trained on SYS1 and SYS2 transcriptions for all BULATS evaluation data sets *BLT-EVLS1-BLT-EVLS3* and *BLT-EVL*. All models achieve a high ROC AUROC scores of over 90.0 on all datasets. Performance of models trained on SYS1 transcriptions and evaluated (on identically constructed) evaluation datasets also based on SYS1 transcripts consistently achieve lower performance than model trained and evaluated on the SYS2 transcripts. However, considering that the ASR performance difference between SYS1 and SYS2 transcriptions is large, as described in table 8.7, the difference in relevance assessment performance is curiously small. The performance on subset *BLT-EVLS1* was highest, which reflects both the dominance of Gujarati L1 candidates in the training data as well as the better quality of the ASR transcriptions of responses of Gujarati candidates. Switching from SYS1 and SYS2 gives a large increase in performance on BLT-EVLS2 and BLT-EVLS3, but gives only a minor gain on *BLT-EVLS1*. This is likely because SYS1 and SYS2 have similar ASR performance on *BLT-EVLS1* data, but difference of nearly 20% WER on *BLT-EVLS2* and *BLT-EVLS3*, as shown in table 8.6. Table 8.8 also

Eval Prompt shuffling	ASR	Evaluation Dataset			
		BLT-EVLS1	BLT-EVLS2	BLT-EVLS3	BLT-EVL
Naive	SYS1	98.5 $\pm$ 0.3	95.5 $\pm$ 0.6	96.0 $\pm$ 0.4	96.8 $\pm$ 0.4
	SYS2	98.4 $\pm$ 0.8	97.4 $\pm$ 1.1	97.2 $\pm$ 1.3	97.7 $\pm$ 1.1
Directed	SYS1	97.0 $\pm$ 0.8	92.5 $\pm$ 1.4	92.5 $\pm$ 1.4	94.2 $\pm$ 1.1
	SYS2	97.1 $\pm$ 1.7	95.6 $\pm$ 2.0	94.9 $\pm$ 2.5	95.9 $\pm$ 2.1

Table 8.8 Mean % AUROC scores $\pm 2\sigma$ on BLT-EVLS1-3 and BLT-EVL across 10 ATM models. Performance is assessed on both SYS1 and SYS2 transcriptions in a matched configuration (training data is also decoded using SYS1 and SYS2). Additionally, sensitivity to prompt shuffling is also assessed.

shows that both models achieve higher performance data with *Naive* topic shuffling than with *Directed* topic shuffling. This supports the findings in [82] which state that it is more difficult to distinguish prompts from the same section than from across sections. Notably the performance difference between SYS1 and SYS2 on BLT-EVLS2 and BLT-EVLS3 is greater on the datasets with *Directed* shuffling. It is likely that having better ASR transcriptions allows the ATM to be able to distinguish prompts within a section more effectively, due to lower ASR noise.

Table 8.9 shows how the AUROC performance varies with the CEFR level of the candidates on subsets of *BLT-EVL* corresponding to speakers with different CEFR grader levels. The main observations is that relevance assessment performance increases with proficiency level. This reflects two effects - firstly, the increasing complexity and clarity of the response,

Eval Prompt Shuffling	CEFR Grade Level A1	A2	B1	B2	C
Naive	93.2 $\pm$ 2.1	96.8 $\pm$ 1.3	98.6 $\pm$ 0.7	98.8 $\pm$ 0.8	98.6 $\pm$ 1.0
Directed	90.4 $\pm$ 3.1	94.5 $\pm$ 2.1	96.8 $\pm$ 1.9	97.6 $\pm$ 1.7	97.3 $\pm$ 2.1

Table 8.9 Per-grade level mean % AUROC scores $\pm 2\sigma$ across 10 ATM models on BLT-EVL using SYS2 transcriptions. In this table the sensitivity of performance to the CEFR grade level of the candidates is assessed.

allowing it to be more easily distinguished from a response to a different prompt, and the rising quality of the transcription, as the ASR performance is higher for better speakers, as described in table 8.7. Interestingly, performance on grade C (highest) speakers is a little lower than on B2 speakers. This reflects the fact that there are fewer speakers of grade C (C1 and C2) than of other grade levels in the training data, as shown in figure 8.7. Given that the experiments in tables 8.8 and 8.9 show that SYS2 transcriptions are better, all further experiments are conducted only on SYS2 transcriptions.

Eval Prompt shuffling	Section C	D	E
Naive	98.8 $\pm$ 0.8	99.5 $\pm$ 0.5	96.8 $\pm$ 1.3
Directed	95.2 $\pm$ 4.3	97.9 $\pm$ 2.0	95.8 $\pm$ 1.7

Table 8.10 Per-section mean % AUROC scores $\pm 2\sigma$ across 10 ATM models on BLT-EVL using SYS2 transcriptions. Here the performance across sections C-E of the BULATS exam and how that is affected by prompt shuffling is assessed.

A breakdown of performance on BLT-EVL per section of the BULATS exam using both *Naive* and *Directed* topic shuffling is presented in table 8.10. The results show that it is easiest to assess performance to section D prompts and hardest on section E prompts. Section D prompts are the most distinct, as they specifically relate to a particular figure or graph, while section E prompts are least distinct, as section E is composed of multi-part prompts, and here each separate sub-prompt is treated separately from all other sub-prompts of the same overall prompt. However, in either case, the model still achieves high performance on all sections. As *Naive* topic shuffling is more representative of real off-topic responses, all further experiments will be done using *Naive* topic shuffling.

8.5.6 Performance on Mismatched Data

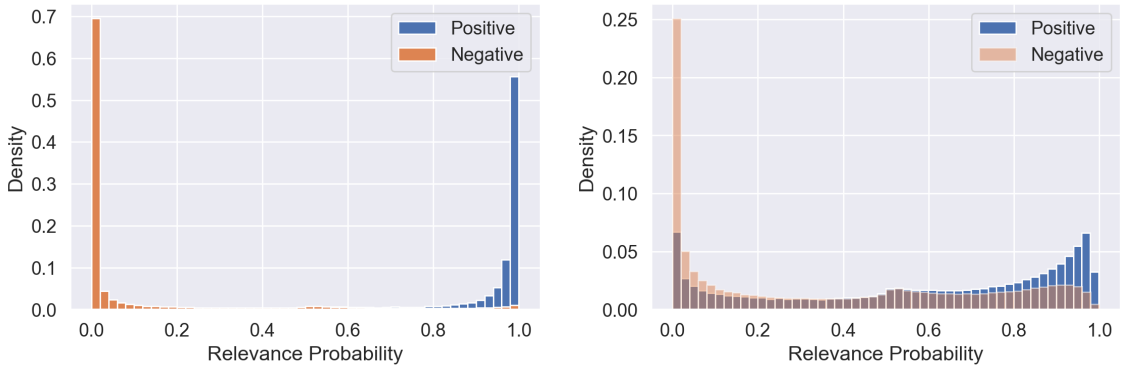
Results in tables 8.8-8.10 show that the ATM (and HATM) model achieves high relevance assessment performance on responses to *known* prompts. However, in sections 8.3 these models are described as being capable of assessing relevance to new and previously *unseen* prompts without retraining the model. Furthermore, the Hierarchical Attention-based Topic model is explicitly introduced in order to improve performance of unseen prompts. In the following experiments, relevance assessment performance on prompt-response pairs from the LinguaSkill spoken language proficiency exam is evaluated.

In the following experiments, described in table 8.11, relevance to prompts which are either *seen* (from BLT-T3) or *unseen* (from LSK) is assessed. Both the HATM and ATM models are only trained on BULATS-derived *BLT-T3* dataset. The evaluation responses are always new (not reused from the training data). However, responses can relate to prompts either seen or unseen in training. Two strategies for shuffling evaluation responses for *negative* examples are considered: *seen*, *unseen*. The first uses responses to seen (BULATS) prompts as negative examples, the second uses responses to unseen (LinguaSkill) prompts as negative examples. This produces four experiments which illustrate different aspects of how well the models understand what relates to seen prompts and how well they generalize to new, unseen prompts. The seen-seen dataset is simply BLT-EVL, the unseen-unseen dataset is LSK, seen-unseen corresponds to having positive examples and negative example *prompts* from BLT-EVL, and negative *responses* from LSK, and unseen-seen takes positive examples and negative example *prompts* from LSK and negative *responses* from BLT-EVL.

Prompts	Negative Responses relating to Seen Prompts		Negative Responses relating to Unseen Prompts	
	ATM	HATM	ATM	HATM
Seen	97.7 $\pm$ 1.0	97.2 $\pm$ 1.3	97.1 $\pm$0.9	96.6 $\pm$1.2
Unseen	63.2 $\pm$6.0	67.8 $\pm$4.2	64.1 $\pm$ 6.3	68.9 $\pm$ 4.5

Table 8.11 Mean % AUROC scores $\pm 2\sigma$ across 10 ATM and HATM models on SYS2 transcriptions. Here Seen-Seen corresponds to BLT-EVL while Unseen-Unseen corresponds to LSK. Seen prompts-unseen is constructed from BLT-EVL, where negative responses are taken from LSK while unseen-seen is LSK with negative responses from BLT-EVL.

Results presented in table 8.11 line 1 show that when both the prompts are seen and when responses relate to seen prompts, both the ATM and HATM have similar, high performance. When the negative responses are taken from a different dataset (seen-unseen), then there is a small drop on performance. This shows that once prompts have been seen in training, the



(a) Seen Seen

(b) Unseen Unseen

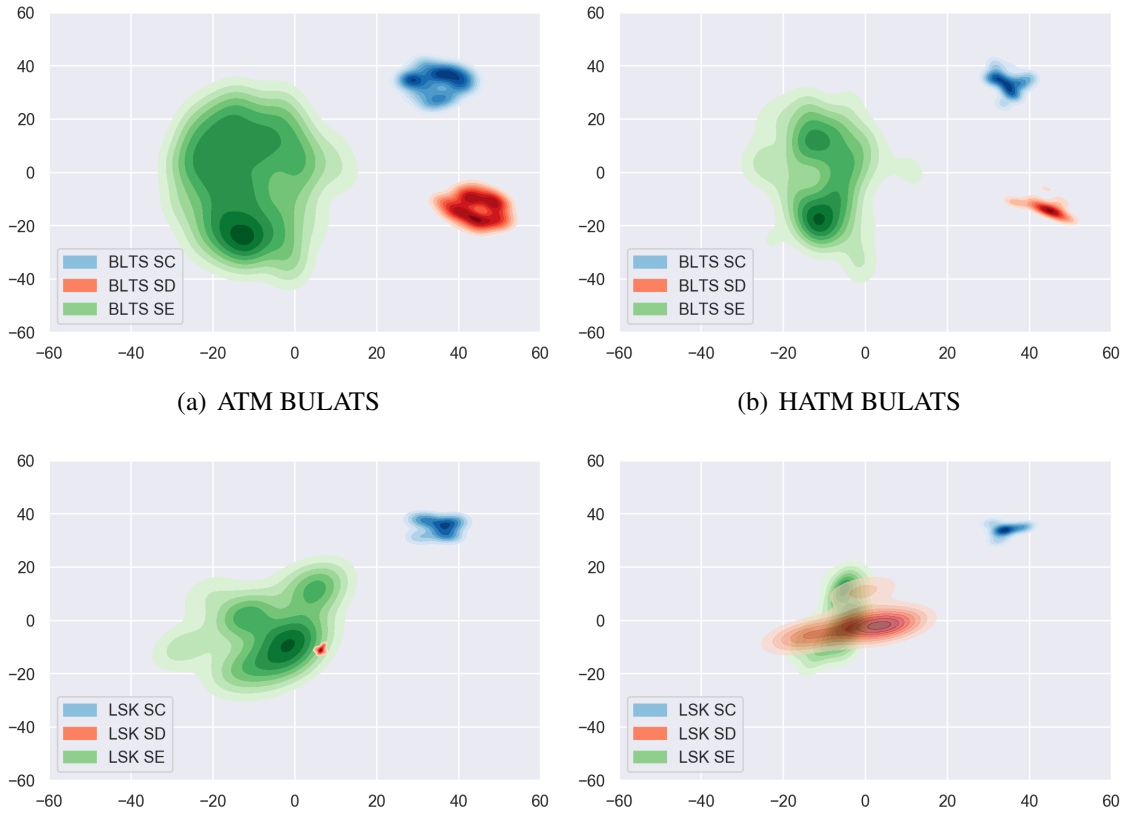
Figure 8.8 Histograms of relevance probability for HATM on BLT-EVL and LSK. Relevance of positive (relevant) and negative (non-relevant) prompt-response pairs is depicted in different. The histograms represent concatenated predictions across all 10 HATM models (with different random seeds).

models have a clear understanding of what is relevant to them and are robust to the nature of the negative-example responses. The situation, however, is worse when assessing relevance to *unseen* prompts. The ROC AUROC drops from being above 90 to being in the mid 60 range. In this scenario, the HATM consistently achieves higher performance than the ATM, although the performance gain is small. This mostly indicates that both the ATM and HATM have overfit to the BULATS prompts, as the training data only contains 379 unique prompts, but greater variety of different responses. Thus, table 8.11 shows that while both models are robust to a diverse set of non-relevant responses, the systems are sensitive to the nature of the prompts. It must be noted that not only are the LSK prompts unseen by the model, but there is also a domain mismatch, as the BULATS prompts are exclusively business related, while the LSK prompts cover a wide range of subjects. Interestingly, the standard deviation of the ROC AUROC on seen-seen data is an order of magnitude smaller than the standard deviation of the ROC AUROC on the LinguaSkill data. This suggests that the LinguaSkill data is out-of-distribution and the ensemble is diverse in that region. This property will be explored in the next section.

It is interesting to analyze the mistakes which the system makes. To do this, the relevance probabilities predicted by an HATM[4] for positive and negative examples are plotted as histograms for the scenarios where seen prompts are used to generate both sets of examples (Fig. 8.8a) and where unseen prompts are used for both (Fig. 8.8b). The other scenarios yield similar histograms. When operating on seen prompts, the model is able to correctly

[4]The HATM is chosen as it has better performance on unseen prompts and essentially identical behaviour to ATM on seen prompts.

classify most examples with very high/low relevance probabilities. However, when operating on unseen prompts it is able to detect when prompts and responses are mismatched, but is unsure about matched prompt-response pairs for unseen prompts, leading to a large number of false-negatives. This is the main failure case of these models. Additionally, in both cases there is a small bump at 0.5, which corresponds to very short responses which the models do not know how to assess. This is an example of *data uncertainty* in relevance assessment. These histograms suggest that the models, via the response attention mechanism, learn a 'lock and key' behaviour, where for a given response, only summation of the hidden states using weights derived from a matched prompt result in a high relevance prediction, and all other summations result in a low relevance prediction. When the prompt-response pair is mismatched, the models generally assign a low relevance probability. However, in the matched case for unseen prompts (LSK) the models yield relevance scores across the whole range from 0.0 to 1.0, which indicates a generalization issue. It should be noted that 'lock and key' behavior reflects the way the models are trained - each response in the training data



(a) ATM BULATS

(b) HATM BULATS

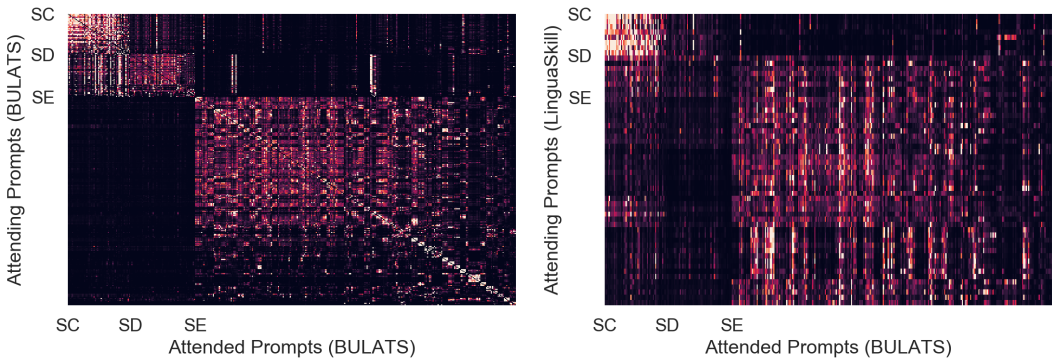
(c) ATM LinguaSkill

(d) HATM LinguaSkill

Figure 8.9 t-SNE projection of prompt embeddings.

is used as a positive example only once, when matched with an appropriate prompt, and many times as a negative example, when matched with any other prompt.

Having established the performance of both the standard discriminative attention topic model and the Hierarchical attention topic model, and seen that the HATM does indeed yield performance benefits on out-of-distribution data, it is interesting to investigate what it has learned. First, t-SNE projections [122] of the prompt embeddings, presented in figure 8.9, are computed on the ATM and HATM for both the BULATS prompts and LinguaSkill prompts. Figures 8.9a and 8.9b show the density plots of the t-SNE projections of the ATM embeddings $\boldsymbol{h}_p$ and attended over HATM embeddings $\boldsymbol{c}_p$ of the BULATS prompts. Both sets of embeddings form three distinct clusters, grouped by section. Notably, the interpolated embeddings $\boldsymbol{c}_p$ reside in the same locations as the originals. Figures 8.9c and 8.9d show the density plots of the projections of the ATM and HATM embeddings of the LinguaSkill prompts. Interestingly, LinguaSkill section C and E prompts map to the BULATS section C and E region, but section D plots map to the section E region. This can be further explored by examining the attention mechanism of the HATM. Figure 8.10 shows a heatmap of the attention of BULATS prompts over BULATS prompts and LinguaSkill Prompts over BULATS prompts. The attention of the BULATS prompts over themselves are zeroed out. Figure 8.10a shows that the prompt attention mechanism clearly learns to distinguish sections - as the attention mechanism mostly focuses on prompts from the same section, leading the to triple-square structure along the diagonal in figure 8.10a. Figure 8.10b shows that the LinguaSkill prompts attend over BULATS section C and E, which supports the observations from figures 8.9. The most likely explanation for this is that LinguaSkill section D prompts



(a) BULATS prompts attending (b) LinguaSkill prompts attending

Figure 8.10 Heatmap visualization of HATM mean attention mechanism across 10 random initialization. X-axis sorted by section (C-E) and by count. Attending prompts are located on the y-axis, wile the prompt being attended over are on the x-axis.

are very distinct from BULATS section C and D prompts, and so are mapped to the most diverse region, which belongs to BULATS section E prompts. Overall, this shows that both the ATM and HATM understand underlying structure of the BULATS and LinguaSkill prompts and cluster them into separate sections. Furthermore, it shows that the prompt attention mechanism is interpretable and allows the behaviour of the HATM model to be analyzed.

8.5.7 Uncertainty for Direct Relevance Assessment

In the previous sections the predictive performance of the ATM and HATM were investigated. It was established that these models are able to achieve high performance on relevance assessment between prompts and responses when the prompts have been seen in training (BULATS prompts). Furthermore, both models are shown to be robust to the nature of the responses. However, the performance of both models drastically falls when assessing relevance to previously unseen prompts with a degree of domain mismatch. Chapter 3 discussed how to derive measures of uncertainty from neural-network based models and chapter 5 showed that it is possible to use measures of uncertainty to detect misclassifications and out-of-distribution inputs. It was shown that explicit ensembles of models, a baseline approach considered in chapter 5, yield both improvement in classification performance and competitive measures of uncertainty. At the same time, it is unclear how to generated out-of-domain training data for relevance assessment, as the input is now a word sequence. Thus, this section investigates using measures of uncertainty, derived from explicit ensemble of HATM models, to do two related tasks. Firstly, running the misclassification detection experiment in the same fashion as described in section 5.3; secondly, using uncertainty to select a subset of data on which to asses relevance, rejecting the rest to human assessors (oracle), similar to the experiments in sections 5.3.3 and 7.2.

As discussed in section 8.3, given a single ATM and HATM model trained with maximum likelihood it is possible to use the entropy of the output distribution as a measure of uncertainty. Additionally, it is possible to use the entropy of the prompt attention mechanism of an HATM as another measure of uncertainty. Finally, given a ensemble of models, it is possible to derive measures of uncertainty such as the entropy of the expected distribution, a measure of *total uncertainty*, as well mutual information and expected pairwise KL-divergence, which are a measures of *knowledge uncertainty* uncertainty. Thus, in this section six measures of uncertainty are investigated - entropy, mutual information, expected pairwise KL-divergence of the output distribution and of the prompt attention mechanism.

Given that measures of uncertainty are derived not only from individual models, but also from an entire ensemble, it is necessary to first compare the performance of an ensemble

to the average performance of each model in the ensemble, as well as the best and worst performance of each model in the ensemble. This comparison is presented in table 8.12. The results show that an ensemble achieves higher performance than the individual models, on average, on both 'in-domain' BLT-EVL data and 'out-of-distribution' LSK data. The best individual model outperforms the ensemble on LSK, but the performance on LSK is sensitive to the initialization - the worst performance on significantly worse than the best.

Data	Ind. Mean	Ind. Best	Ind. Worst	Ensemble
BLT-EVL	97.5	97.7	95.7	98.2
LSK	68.9	73.4	65.2	71.6

Table 8.12 Comparison of Individual model and Ensemble performance of HATM on BLT-EVL and LSK datasets in terms of % AUROC.

Table 8.13 presents the results of the misclassification experiment conducted on both the individual models as well as the ensemble. Here, the classification is made at a decision threshold of 0.5, as it represents a 2-class application of the argmax decision rule. The results show that the error (misclassification) rate on BLT-EVL is significantly lower than on LSK. It is important to know the error rate for two reasons - it represents the 'random' performance on a PR curve and also it shows the misbalance of the dataset (correct - incorrect classifications). If the dataset is heavily misbalanced, AUPR will be a better indicator of performance than AUROC. Measures of uncertainty derived from the output are computed using equations 3.31 and 3.32, while measures of uncertainty derived from the prompt attention mechanism are obtained via 8.31 and 8.32.

Table 8.13 shows that it is possible to detect misclassifications on both BLT-EVL and LSK datasets. Misclassification detection on LSK data is more difficult - it is out-of-distribution relative to the training data and models may yield poor estimates of uncertainty in this region. Note, the difference between AUPR and Error is larger on BLT-EVL than on LSK. Table 8.13 also shows that measures of *total uncertainty*, such as entropy of the output or the prompt attention mechanism, are best for misclassification detection. At the same time measures of *knowledge uncertainty*, such as mutual information and expected pairwise KL-divergence, are worse for detecting misclassifications. This is in agreement with results from chapter 5. Notably, measures of uncertainty derived from the prompt attention mechanism are only marginally worse than measures derived from the output on BLT-EVL. At the same time, they are better than measures of uncertainty derived from the output on LSK. This suggests that most of the uncertainty in predictions is due to the prompt, rather than response, which is supported by results from section 8.5.5. A significant advantage of measures of uncertainty derived from the prompt attention mechanism is that they do not rely of the response to be

Data	Type	Output			Prompt Attention			% Error
		Ent	MI	EPKL	P.Ent	P.MI	P.EPKL	
BLT-EVL	IND	38.3 $\pm$ 1.3	-	-	34.5 $\pm$ 2.3	-	-	8.9
	ENS	**38.7** $\pm$ NA	18.9 $\pm$ NA	17.9 $\pm$ NA	30.9 $\pm$ NA	18.4 $\pm$ NA	17.6 $\pm$ NA	7.3
LSK	IND	45.5 $\pm$ 2.6	-	-	47.4 $\pm$ 3.7	-	-	37.1
	ENS	47.6 $\pm$ NA	50.9 $\pm$ NA	51.1 $\pm$ NA	50.8 $\pm$ NA	**51.3** $\pm$ NA	50.9 $\pm$ NA	36.7

Table 8.13 Misclassification detection experiment in terms of % AUPR ($\pm 2\sigma$ on individual models).

computed. Thus, an organization which plans to add new prompts to an exam can know prior to deployment whether the system will perform well on the prompt or not. This is a significant operational advantage.

In addition to the misclassification detection, measures of uncertainty derived from an HATM were evaluated via prediction rejection experiments in the same fashion as described in section 5.2. The rejection curves are shown in figure 8.11 and rejection ratios RR are presented in table 8.14. In this experiment, predictions of the HATM are rejected in order of decreasing uncertainty and replaced with the true labels. This models the scenario where the model backs-off to a human assessor when it is uncertain. The expected rejection curve for a random rejection would be a straight line between base performance and the top right corner. The 'oracle' curve represents rejection in order of decreasing error with the label. In a real situation, this curve is unavailable, but represents asking an oracle, in this scenario, a human assessor, to provide the correct prediction. If a rejection scheme is perfect, it should follow the oracle curve. Figure 8.11 shows that on the estimates of uncertainty provided by the model can be successfully used to reject some of the worst responses. Note, that the provided measures of uncertainty are better for in-domain BLT-EVL data than for LSK data. Notably, entropy of the output and of the prompt attention mechanism is the best measures

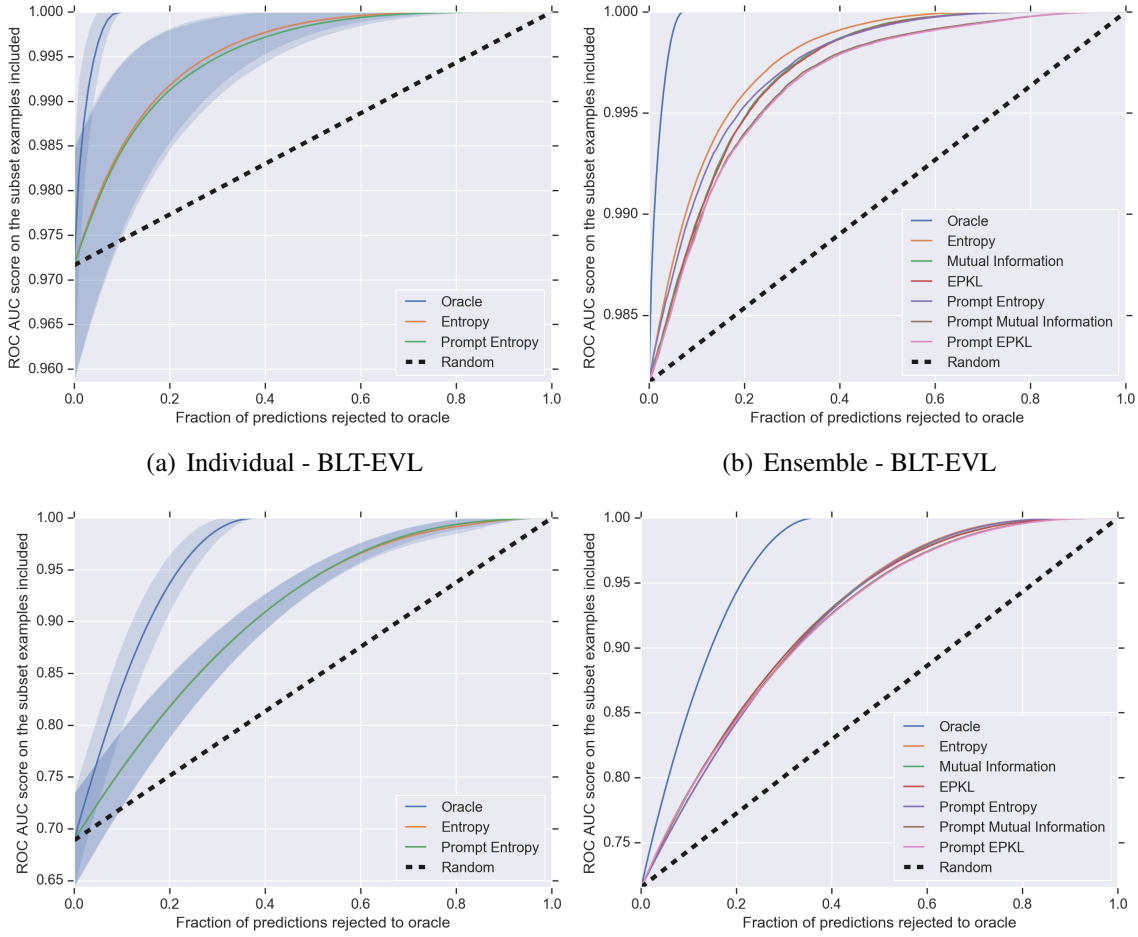
Data	Type	Output			Prompt Attention		
		Ent	MI	EPKL	P.Ent	P.MI	P.EPKL
BLT-EVL	IND	72.6 $\pm$ 5.8	-	-	70.5 $\pm$ 6.5	-	-
	ENS	**77.0** $\pm$ NA	71.1 $\pm$ NA	70.6 $\pm$ NA	73.5 $\pm$ NA	65.8 $\pm$ NA	65.0 $\pm$ NA
LSK	IND	54.4 $\pm$ 4.3	-	-	54.7 $\pm$ 4.0	-	-
	ENS	61.6 $\pm$ NA	**62.1** $\pm$ NA	61.9 $\pm$ NA	61.2 $\pm$ NA	60.1 $\pm$ NA	59.7

Table 8.14 Prediction rejection ratios RR on BLT-EVL and LSK. For individual models this is a mean $\pm 2\sigma$ across 10 models.

of uncertainty on in-domain BLT-EVL data. However, on LSK the situation is less clear cut, where all measures have similar performance. The results show that measures derived from the output are nearly as good as measures of uncertainty derived from the prompt attention mechanism. This is a significant operational advantage in a deployment scenario - if a new prompt is introduced into an exam, it is possible to predict expected relevance assessment performance on that prompt and decide whether it is necessary to collect more training data.

8.6 Chapter Summary

This chapter investigates the assessment of relevance between prompts and spoken responses. Two approaches to prompt-response relevance assessment were considered - an indirect



(a) Individual - BLT-EVL

(b) Ensemble - BLT-EVL

(c) Individual - LSK

(d) Ensemble - LSK

Figure 8.11 Prediction rejection curves on BLT-EVL and LSK. For individual models (a and c) the mean rejection curve $\pm 2\sigma$ error bounds across 10 models are depicted.

approach, discussed in section 8.2, where a model predicts which prompt most likely elicited the given response, and a direct approach, discussed in section 8.3, where a model directly yields a relevance score for a prompt-response pair.

In section 8.2 two indirect prompt-relevance assessment models were considered. The first is a classic information-retrieval style approach which relies on cosine distances between vector representations of prompts and responses combined with a K-nearest neighbour classifier. The second is a deep-learning based prompt-adapted RNN language model, which is a generative model of responses, conditioned on a vector embedding of the prompt. Conditional probabilities of the prompt are obtained by applying Bayes' rule. Experiments in section 8.4 show that the second model is able to scale to larger datasets, as it is parametric, and yields better relevance assessment performance. However, a limitation of this approach is that it still requires running the RNNLM as many times as there are prompts in the dataset for each response, which can become expensive. Furthermore, these indirect approaches cannot handle new and unseen prompts without retraining.

In section 8.3 two direct prompt-relevance assessment models, the Attention-based Model (ATM) and the Hierarchical Attention-based Model (HATM) are considered. Both models use a prompt-conditional attention mechanism is attend over embeddings of responses, produced by a response-encoder, to select only the most salient parts of the response for relevance assessment. As the models also feature a prompt-encoder, they are theoretically capable as assessing relevance between *arbitrary* prompt-response pairs without retaining. The HATM represents all input prompts as a convex combination of prompts seen in the training data via a second *prompt attention mechanism*. This allows the HATM to yield both more robust and *interpretable* representations of prompts.

The direct relevance assessment models were experimentally evaluated on data from the BULATS and LinguaSkill spoken language proficiency exams in section 8.5. Results show that when assessing relevance of responses to known prompts, these models achieve very high, near perfect performance. However, when assessing relevance to new and unseen prompts, these models yields lower performance. However, the HATM, with its prompt attention mechanism, does achieve better performance than the ATM. The reason for this is that the training data contains very few unique prompts and the models overfit to them, making it difficult to generalize to new prompts. At the same time, it is shown that both models are robust to the nature of responses. Furthermore, it is shown that the prompt-attention mechanism is interpretable and yields insights into the behaviour of the HATM model. Specifically, it is shown that the prompt-attention mechanism learns to cluster prompts by BULATS-exam sections in an unsupervised fashion. Future work must investigate approaches

to enhancing performance on unseen prompts. One approach would be to use the diversity of available responses to train the prompt-encoder on mixture of both prompts and responses.

Section 8.5.7 investigates deriving measures of uncertainty from ensembles of HATM models for two tasks: misclassification detection and uncertainty-based prediction rejection. The results show that estimates of model's uncertainty can be applied to both tasks successfully. Crucially, it was shown that useful measures of uncertainty can be derived from the prompt-attention mechanism for both misclassification detection and uncertainty-based rejection. This means that a level of expected performance on the prompt can be obtained based on the prompt alone, which is a significant operational advantage, as it allows the model's constructor to decide when to collect more training responses and when no new training responses are necessary for a new prompt. This should yield a decrease in operations cost.

However, the investigated approaches to deriving measures of uncertainty are computationally expensive. Future work should investigate combination of HATM models with Prior Networks, in order to derive computationally cheaper measures of uncertainty and more accurate predictions. An important step towards this is the investigation of out-of-distribution training data generation for structured inputs, such as word sequences. Furthermore, the usefulness of measures of uncertainty derived from the prompt-attention mechanism suggests that Prior Networks over *intermediate* categorical representations inside a neural network should also be investigated in future work.

Chapter 9

Conclusions

The current chapter concludes this thesis with a review of the main contributions and several proposals for future work. The main theme of this thesis has been the investigation of approaches for deriving measures of uncertainty in the predictions of deep learning models. This thesis has two main areas of contributions. The first is the development and evaluation of a new class of models, called Prior Networks, for uncertainty estimation which combine properties of single model and ensemble approaches. Effectively, Prior Networks can efficiently emulate an ensemble of either classification or regression models using a single neural network. The second area of contribution is the application of deep learning and uncertainty estimation techniques discussed in this thesis to the area of non-native spoken language proficiency assessment. Specifically, DNNs are applied to the tasks of automatic grading of spoken language proficiency exams and automatic assessment of relevance of spoken responses to open-ended exam prompts. Ensemble approaches were then used to derive measures of uncertainty from deep-learning models for grading and relevance assessment. A per-chapter breakdown of these contributions and a discussion of future research directions is presented in this chapter.

9.1 Review of Contributions

Chapter 3 introduced the area of uncertainty estimation for parametric models. The sources of uncertainty in predictions are discussed in the context of both classification and regression tasks and illustrated on toy artificial datasets. It is illustrated how uncertainty in predictions can arise due to *data uncertainty*, which is uncertainty due to class overlap and noise in the data, and due to *knowledge uncertainty*, which is uncertainty due to the mismatch between the training and test data distributions. This chapter shows that standard probabilistic classification and regression models will naturally capture estimates of *data uncertainty*

if trained using maximum likelihood, given sufficient training data and model capacity. However, standard maximum likelihood estimation does not contain any mechanism to capture *knowledge uncertainty*.

Chapter 3 goes on to discuss two classes of approaches to deriving estimates of *knowledge uncertainty* in the predictions of parametric classification and regression models. The first class is single model approaches, which aim to *explicitly* train a DNN to yield high-entropy outputs via multi-task training on out-of-distribution data. While these approaches are computationally cheap and allow the model to indicate that it is uncertain, they do not allow the model to indicate *why* it is uncertainty in a non-heuristic fashion. This makes it difficult to use these measures of uncertainty for tasks which require the source of uncertain to be established. Instead of single-model approaches, it is possible to consider *ensemble approaches*, which yield interpretable measures of uncertainty within a theoretically consistent probabilistic framework. By considering measures of spread of ensembles of models, measures of *knowledge uncertainty* and *data uncertainty* can be separately derived. However, it is hard to control the behavior of an ensemble of models. Furthermore, ensemble approaches may be computationally expensive.

In chapter 4 a new class of models for uncertainty estimation, called Prior Networks, is proposed. Prior Networks combine the advantages of single-model and ensemble approaches to uncertainty estimation. Specifically, Prior Networks parameterize conjugate prior distributions over output distributions and yield the same theoretically interpretable measures of uncertainty as ensembles. However, unlike ensembles, all measures of uncertainty can be obtained in closed form using a single deterministic forward pass through a neural network. Furthermore, unlike ensembles, whose behaviour is *implicitly* controlled by choice of prior over models and approximate inference scheme, the behaviour of Prior Networks is controlled *explicitly* via the choice of out-of-distribution training data. In this chapter Prior Networks are developed both for classification tasks, where they parameterize a Dirichlet distribution, and for regression tasks, where they parameterize the Normal-inverse-Wishart distribution. In sections 4.2.2 and 4.3.2 training criteria for Prior Networks are investigated and it is shown that in order to appropriately train a Prior Network to yield a desired set of behaviours it is necessary to minimize the *reverse* KL-divergence between a target prior distribution and the Prior Networks. The properties of the reverse KL-divergence loss are confirmed by training Prior Networks on two artificial datasets in section 4.2.3.

Having confirmed the properties of Prior Networks on two artificial datasets, Prior networks are evaluated on the MNIST, SVHN and CIFAR-10 image classification datasets in chapter 5. Uncertainty estimates derived from Prior Networks are applied to the tasks of misclassification detection and out-of-distribution input detection. Standard DNNs, Monte-

Carlo Dropout Ensembles and explicit ensembles of DNNs are used as the baselines in this set of experiments. In section 5.3 it is shown that Prior Networks do not yield improved misclassification detection performance, most likely due to limitations in the out-of-distribution training data. However, they are shown to obtain marginally better classification performance than standard DNNs, most likely due to the out-of-distribution data acting as a regularizer. The best misclassification detection performance is obtained using measures of uncertainty derived from explicit ensemble of DNNs. Furthermore, it is shown how the confidence, also known as the probability of the mode of the predictive posterior, is consistently the best measure of uncertainty for misclassification detection.

At the same time, in section 5.4 it is shown how Prior Networks, given an appropriate choice of out-of-distribution training data, are capable of outperforming the baselines approaches on the task of out-of-distribution input detection. Prior Networks consistently yielded the best results on all datasets. This is likely due to the fact that it is easier to choose out-of-distribution training data which is good for discriminating between the in-domain data and a range of out-of-domain image datasets. The best baseline model was an explicit ensemble of DNNs, while Monte-Carlo dropout ensembles did not give significant benefits over deterministic DNNs. On this task it was shown the measures of *knowledge uncertainty*, such as mutual information and expected pairwise KL-divergence yield the best results.

Chapter 6 transitions from the first part of this thesis, which examined uncertainty estimation, to the second part, which investigates the application of deep learning and uncertainty estimation to the area of spoken language proficiency assessment.

In chapter 7 probabilistic regression models, called Density Networks, were applied to the task of automatic grading based on a set of features derived from the audio and ASR transcriptions of a candidate's responses to open-ended prompts on the BULATS exam. The two classes of approaches to capturing *knowledge uncertainty* discussed in chapter 3 were applied to Density Networks. Specifically, Density Networks were trained in a multi-task fashion on synthetically generated out-of-distribution training data to yield high-entropy predictive posteriors; Monte-Carlo dropout and explicit ensembles of Density Networks were used; finally, ensemble and single-model approaches were combined by considering an explicit ensemble of multi-task trained Density Networks. The previous state-of-the-art model on the this task was a Gaussian process, which was used as the baseline model in this chapter. Experiments in section 7.2 show that all models achieve comparable human-level predictive performance, however, explicit ensembles of multi-task trained Density Networks obtained the best performance. However, the experiments also show that unlike for the classification tasks considered in chapter 5, the measures of uncertainty derived from all models yield low performance of the task of rejecting predictions which are most errorful.

However, considering that models operating on different sets of principles yield similarly low performance suggests that either the features are too limited, or the errors are dominated by homoscedastic (input-independent) noise. At the same time, calibration experiments show that all models yield well-calibrated predictive intervals.

In Chapter 8 deep-learning approaches were applied to the task of automatic assessment of relevance of spoken responses to open-ended exam prompts based on speech-recognition hypotheses. Two classes of approaches were discussed: *indirect* approaches, which assess relevance via prompt classification, and *direct* approaches, which directly yield a relevance score for a prompt-response pair. Two indirect prompt-relevance assessment models were considered - a classic information-retrieval style approach which relies on cosine distances between vector representations of prompts and responses, and a deep-learning based prompt-adapted RNN language model. The latter is a generative model of responses, conditioned on a vector embedding of the prompt. Conditional probabilities of the prompt are obtained by applying Bayes' rule. Experiments on data from the BULATS exam in section 8.4 show that the deep-learning based model is able to scale to larger datasets and yields better performance. However, indirect relevance assessment approaches cannot handle new and unseen prompts without retraining.

Two direct prompt-relevance assessment models, the Attention-based Model (ATM) and the Hierarchical Attention-based Model (HATM) are considered in section 8.3. Both models use bi-directional LSTM embeddings of prompts to attend over bi-directional LSTM embeddings of responses to assess relevance, which allows them to assess relevance between *arbitrary* prompt-response pairs without retraining. Furthermore, the HATM aims to improve generalization to unfamiliar prompts by expressing any input prompts as a convex combination of the embeddings of prompts seen in the training data via a *prompt attention mechanism*. Experiments on data from the BULATS and LinguaSkill spoken language proficiency exams, provided by Cambridge English Language Assessment, show that when assessing relevance of responses to known prompts, these models achieve very high, near perfect performance. However, when assessing relevance to new and unseen prompts, these models yield lower performance. However, the HATM achieves marginally better performance than the ATM. Section 8.5.7 investigates deriving measures of uncertainty from ensembles of HATM models for two tasks: misclassification detection and uncertainty-based prediction rejection. It is shown that measures of uncertainty derived either from the outputs or the prompt attention mechanism of the HATM can be successfully used to detect misclassifications and reject predictions to human assessors. Crucially, the fact that the prompt attention mechanism yields useful measures of uncertainty is a significant operational advantage, as it means that

when a new prompt is introduced into an exam, it is possible to see whether the model will perform well on it without needing actual responses to this prompt.

9.2 Future Work

A range of theoretical extensions of Prior Networks should be investigated. Firstly, results in chapter 3 show that the quality of uncertainty estimates from Prior Networks depends heavily on the nature of the out-of-distribution training data. Thus, it is important to thoroughly investigate approaches to generating or collecting out-of-distribution training data. Secondly, since explicit ensembles outperform Prior Networks on the task of misclassification detection and Prior Networks are capable of emulating an ensemble, an interesting avenue of investigation is the *distillation* of an ensemble into a Prior Network. Thirdly, this thesis investigated the Prior Networks for discriminative models with a vector to vector mapping. As a future avenue of research, it is necessary to investigate derivation of measures of uncertainty for structured data, such as sequences, lattices, graphs and prefix trees in order to be able to apply approaches like Prior Networks to a wide range of real tasks which operate on structured data, such as speech recognition, machine translation, syntactic parsing, etc... Finally, in chapter 8 it was shown that useful measures of uncertainty can be derived from intermediate representations in a neural networks, specifically a fixed-length attention mechanism. Thus, an interesting direction for future research is the construction of Prior Networks over intermediate representations inside neural networks.

In addition to theoretical extension of Prior Networks, several experimental extensions to work on Prior Networks should also be carried out. Firstly, classification Prior Networks should be to far larger and more complex image classification datasets such as ImageNet [27]. Secondly, it is necessary to investigate the use of Prior Networks for different areas of application, such natural language processing and reinforcement learning. Thirdly, it is necessary to experimentally evaluate regression Prior Networks on a range of both synthetic and real tasks. Finally, it is interesting to investigate the use of measures of uncertainty derived from Prior Networks on the task of adversarial attack detection.

Several avenues of future work can also be outlined for the area of automatic spoken language proficiency assessment. Firstly, it is necessary to use a regression Prior Network for automatic grading instead of the Density Networks considered in chapter 7. Secondly, classification approaches to automatic grading should also be investigated. Additionally, it is interesting to investigate the training automatic graders on much larger corpora and on richer features. With regards to prompt-response relevance assessment, it is necessary to investigate approaches to improving performance on unseen prompts. This can either be done

by considering more advanced architectures, leveraging the availability of a diverse range of responses to prompts to train prompt embeddings or by using transfer learning approaches.

References

[1] Computer says no: Irish vet fails oral english test needed to stay in australia. *The Guardian*, 2017. URL https://www.theguardian.com/australia-news/2017/aug/08/computer-says-no-irish-vet-fails-oral-english-test-needed-to-stay-in-australia.

[2] Linguaskill. *Cambridge English Language Assessment*, 2019. URL https://www.cambridgeenglish.org/exams-and-tests/linguaskill/.

[3] Malinin A, K. Knill, A. Ragni, Y. Wang, and M.J.F. Gales. An attention based model for off-topic spontaneous spoken respnse detection: An Initial Study. In *Proc. ISCA Workshop on Speech and Language Technology for Education (SLaTE)*, 2017.

[4] Martín Abadi et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. URL http://tensorflow.org/. Software available from tensorflow.org.

[5] Babak Alipanahi, Andrew Delong, Matthew T. Weirauch, and Brendan J. Frey. Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nature Biotechnology*, 33(8):831–838, July 2015. ISSN 1087-0156. doi: 10.1038/nbt.3300. URL http://dx.doi.org/10.1038/nbt.3300.

[6] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. http://arxiv.org/abs/1606.06565, 2016. arXiv: 1606.06565.

[7] REINALDO B ARELLANO-VALLE, JAVIER E CONTRERAS-REYES, and Marc G Genton. Shannon entropy and mutual information for multivariate skew-elliptical distributions. *Scandinavian Journal of Statistics*, 40(1):42–62, 2013.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proc. International Conference on Learning Representations (ICLR)*, 2015.

[9] Yoshua Bengio. Learning Deep Architectures for AI. *Found. Trends Mach. Learn.*, 2 (1):1–127, 2009.

[10] C. M. Bishop. Mixture Density Networks. *Technical Report NCRG 4288, Neural Computing Research Group, Department of Computer Science, Aston University*, 1994.

[11] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[12] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, March 2003.

[13] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

[14] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.

[15] BULATS. Business Language Testing Service. http://www.bulats.org, 2012.

[16] M Buscema. Metanet: The theory of independent judges. *Substance Use & Misuse*, 33(2):439–461, 1998.

[17] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proc. 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1721–1730, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3664-2. doi: 10.1145/2783258.2788613. URL http://doi.acm.org/10.1145/2783258.2788613.

[18] Lucy Chambers and Kate Ingham. The BULATS online speaking test. *Research Notes*, 43:21–25, 2011.

[19] X. Chen, X. Liu, Y. Qian, M.J.F. Gales, and P.C. Woodland. CUED-RNNLM – An Open-Source Toolkit for Efficient Training and Evaluation of Recurrent Neural Network Language Models. In *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2016.

[20] Xie Chen, Yongqiang Wang, Xunying Liu, Mark J.F. Gales, and P.C. Woodland. Efficient GPU-based Training of Recurrent Neural Network Language Models Using Spliced Sentence Bunch. In *Proc. INTERSPEECH*, 2014.

[21] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[22] Robert J Connor and James E Mosimann. Concepts of independence for proportions with a generalization of the dirichlet distribution. *Journal of the American Statistical Association*, 64(325):194–206, 1969.

[23] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2006.

[24] Stanford CS231N. Tiny ImageNet. https://tiny-imagenet.herokuapp.com/, 2017.

[25] C. Cucchiarini, H. Strik, and L. Boves. Automatic evaluation of Dutch pronunciation by using speech recognition technology. In *Proc. of IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, pages 622–629, 1997.

[26] M Del Vecchio, A Malinin, and MJF Gales. Improved auto-marking confidence for spoken language assessment. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 957–963. IEEE, 2018.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[28] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Decomposition of uncertainty for active learning and reliable reinforcement learning in stochastic systems. *stat*, 1050:11, 2017.

[29] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul): 2121–2159, 2011.

[30] Keelan Evanini and Xinhao Wang. Automatic detection of plagiarized spoken responses. In *Proc. Ninth Workshop on Innovative Use of NLP for Building Educational Applications*, 2014.

[31] Keelan Evanini, Shasha Xie, and Klaus Zechner. Prompt-based Content Scoring for Automated Spoken Language Assessment. In *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2013.

[32] G. Evermann and P.C. Woodland. Large vocabulary decoding and confidence estimation using word posterior probabilities. In *Proc. of IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2000.

[33] H. Franco, V. Abrash, K. Precoda, H. Bratt, R. Rao, J. Butzberger, R. Rossier, and F. Cesari. The SRI EduSpeakTM system: Recognition and pronunciation scoring for language learning. *Proc. of InSTILL 2000*, pages 123–128, 2000.

[34] H. Franco, L. Neumeyer, V. Digalakis, and O. Ronen. Combination of machine scores for automatic grading of pronunciation quality. *Speech Communication*, 30(2): 121–130, 2000.

[35] Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

[36] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proc. 33rd International Conference on Machine Learning (ICML-16)*, 2016.

[37] Ross Girshick. Fast R-CNN. In *Proc. 2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[38] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

[39] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[40] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[41] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay D. Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks, 2013. URL http://arxiv.org/abs/1312.6082. arXiv:1312.6082.

[42] Alex Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*. Studies in Computational Intelligence, Springer, 2012.

[43] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural Turing Machines, 2014. URL http://arxiv.org/abs/1410.5401. arXiv:1410.5401.

[44] Thomas L. Griffiths and Mark Steyvers. Finding Scientific Topics. *Proceedings of the National Academy of Sciences*, 101:5228–5235, 2004.

[45] Sylvain Gugger. The 1-cycle policy. 2018. URL https://sgugger.github.io/the-1cycle-policy.html.

[46] Maya Gupta and Santosh Srivastava. Parametric bayesian estimation of differential entropy and relative entropy. *Entropy*, 12(4):818–843, 2010.

[47] Awni Y. Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition, 2014. URL http://arxiv.org/abs/1412.5567. arXiv:1412.5567.

[48] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[49] Dan Hendrycks and Kevin Gimpel. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. http://arxiv.org/abs/1610.02136, 2016. arXiv:1610.02136.

[50] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701, 2015.

[51] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

[52] Derrick Higgins, Xiaoming Xi, Klaus Zechner, and David Williamson. A three-stage approach to the automated scoring of spontaneous spoken responses. *Computer Speech and Language*, 25(2):282–306, 2011.

[53] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine*, 2012.

[54] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. arXiv:1503.02531.

[55] Geoffrey E. Hinton and Drew van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proc. Sixth Annual Conference on Computational Learning Theory*, COLT '93, pages 5–13, New York, NY, USA, 1993. ACM. ISBN 0-89791-611-5. doi: 10.1145/168304.168306. URL http://doi.acm.org/10.1145/168304.168306.

[56] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. http://arxiv.org/abs/1207.0580, 2012. arXiv:1207.0580.

[57] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8): 1735–1780, 1997.

[58] Marco F Huber, Tim Bailey, Hugh Durrant-Whyte, and Uwe D Hanebeck. On entropy approximation for gaussian mixture random vectors. In *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 181–188. IEEE, 2008.

[59] Forrest Iandola, Matt Moskewicz, Sergey Karayev, Ross Girshick, Trevor Darrell, and Kurt Keutzer. Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869*, 2014.

[60] A. Kendall and Y. Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision. In *Proc. Conference on Neural Information Processing Systems (NIPS)*, 2017.

[61] A. Kendall, Y. Gal, and R. Cipolla. Multi-Task Learning Using Uncertainty to Weight Losses for Scene Geometry and Semantics. In *Proc. Conference on Neural Information Processing Systems (NIPS)*, 2017.

[62] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proc. 3rd International Conference on Learning Representations (ICLR)*, 2015.

[63] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *Proc. International Conference on Learning Representations (ICLR)*, 2014.

[64] Artemy Kolchinsky and Brendan Tracey. Estimating mixture entropy with pairwise distances. *Entropy*, 19(7):361, 2017.

[65] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[66] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[67] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. ISSN 0036-8075. doi: 10.1126/science.aab3050. URL http://science.sciencemag.org/content/350/6266/1332.

[68] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *Proc. Conference on Neural Information Processing Systems (NIPS)*, 2017.

[69] Thomas K Landauer, Peter W. Foltz, and Darrell Laham. Introduction to Latent Semantic Analysis. *Discourse Processes*, 25:259–284, 1998.

[70] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the* IEEE, 86:2278–2324, 1998.

[71] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[72] Chong Min Lee, Su-Youn Yoon, Xihao Wang, Matthew Mulholland, Ikkyu Choi, and Keelan Evanini. Off-topic spoken response detection using siamese convolutional neural networks. In *INTERSPEECH*, pages 1427–1431, 2017.

[73] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=ryiAv2xAZ.

[74] Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *Proc. International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1VGkIxRZ.

[75] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 2017.

[76] Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. Hierarchical question-image co-attention for visual question answering. In *Advances In Neural Information Processing Systems*, pages 289–297, 2016.

[77] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proc. EMNLP*, 2015.

[78] David JC MacKay. *Bayesian methods for adaptive models*. PhD thesis, California Institute of Technology, 1992.

[79] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[80] A. Malinin, A. Ragni, M.J.F. Gales, and K.M. Knill. Incorporating Uncertainty into Deep Learning for Spoken Language Assessment. In *Proc. 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

[81] Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems*, pages 7047–7058, 2018.

[82] Andrey Malinin, Rogier van Dalen, Kate Knill, Yu Wang, and Mark Gales. Off-topic Response Detection for Spontaneous Spoken English Assessment. In *Proc. 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1075–1084, Berlin, Germany, 2016.

[83] Andrey Malinin, Kate Knill, and Mark JF Gales. A hierarchical attention based model for off-topic spontaneous spoken response detection. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 397–403. IEEE, 2017.

[84] R Mead. A generalised logit-normal distribution. *Biometrics*, 21(3):721–732, 1965.

[85] Angeliki Metallinou and Jian Cheng. Using Deep Neural Networks to Improve Proficiency Assessment for Children English Language Learners. In *Proc. INTERSPEECH*, 2014.

[86] Tomas Mikolov. *Statistical Language Models Based on Neural Networks*. PhD thesis, Brno University of Technology, 2012.

[87] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent Neural Network Based Language Model. In *Proc. INTERSPEECH*, 2010.

[88] Tomas Mikolov, Stefan Kombrink, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. Extensions of Recurrent Neural Network Language Model. In *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2011.

[89] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space, 2013. URL http://arxiv.org/abs/1301.3781. arXiv:1301.3781.

[90] Tomas Mikolov et al. Linguistic Regularities in Continuous Space Word Representations. In *Proc. NAACL-HLT*, 2013.

[91] Marcin Możejko, Mateusz Susik, and Rafał Karczewski. Inhibited softmax for uncertainty estimation in neural networks. *arXiv preprint arXiv:1810.01861*, 2018.

[92] Kevin P. Murphy. *Machine Learning*. The MIT Press, 2012.

[93] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, 2016.

[94] Radford M. Neal. *Bayesian learning for neural networks*. Springer Science & Business Media, 1996.

[95] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[96] Council of Europe. *Common European framework of reference for languages: Learning, teaching, assessment*. Cambridge, U.K: Press Syndicate of the University of Cambridge, 2001. ISBN 9780521005319.

[97] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

[98] Ian Osband. Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout. In *NIPS Workshop on Bayesian Deep Learning*, 2016.

[99] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4026–4034. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6501-deep-exploration-via-bootstrapped-dqn.pdf.

[100] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[101] Xuan-Hieu Phan and Cam-Tu Nguyen. GibbsLDA++: A C/C++ implementation of latent Dirichlet allocation (LDA). http://gibbslda.sourceforge.net/, 2007.

[102] Joaquin Quiñonero-Candela. *Dataset Shift in Machine Learning*. The MIT Press, 2009.

[103] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[104] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

[105] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[106] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[107] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[108] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[109] Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, 2017.

[110] Barbara Seidlhofer. English as a lingua franca. *ELT journal*, 59(4):339, 2005.

[111] Murat Sensoy, Lance Kaplan, and Melih Kandemir. Evidential deep learning to quantify classification uncertainty. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 3179–3189. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7580-evidential-deep-learning-to-quantify-classification-uncertainty.pdf.

[112] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[113] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proc. International Conference on Learning Representations (ICLR)*, 2015.

[114] Leslie N Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018.

[115] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. *arXiv preprint arXiv:1708.07120*, 2017.

[116] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[117] Nitish Srivastava et al. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[118] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.

[119] Rogier C. van Dalen, Kate M. Knill, and Mark J. F. Gales. Automatically Grading Learners' English Using a Gaussian Process. In *Proc. ISCA Workshop on Speech and Language Technology for Education (SLaTE)*, 2015.

[120] Rogier C. van Dalen, Kate M. Knill, Pirros Tsiakoulis, and Mark J. F. Gales. Improving Multiple-Crowd-Sourced Transcriptions Using a Speech Recogniser. In *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2015.

[121] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in neural information processing systems*, pages 4790–4798, 2016.

[122] L. van der Maaten and G. Hinton. Visualizing Data using t-SNE. *J. MLR*, 1:1–49, 2008.

[123] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[124] N. Verhelst et al. *Common European Framework of Reference for Languages: learning, teaching, assessment*. Cambridge University Press, 2009.

[125] Ruben Villegas, Jimei Yang, Yuliang Zou, Sungryull Sohn, Xunyu Lin, and Honglak Lee. Learning to Generate Long-term Future via Hierarchical Prediction. In *Proc. International Conference on Machine Learning (ICML)*, 2017.

[126] Yu Wang, MJF Gales, Katherine Mary Knill, K Kyriakopoulos, Andrey Malinin, RC van Dalen, and M Rashid. Towards automatic assessment of spontaneous spoken english. *Speech Communication*, 104:47–56, 2018.

[127] Max Welling and Yee Whye Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *Proc. International Conference on Machine Learning (ICML)*, 2011.

[128] S. M. Witt. *Use of speech recognition in computer-assisted language learning*. PhD thesis, University of Cambridge, 1999.

[129] S. M. Witt and S. J. Young. Phone-level pronunciation scoring and assessment for interactive language learning. *Speech Communication*, 30(2):95–108, 2000.

[130] Tzu-Tsung Wong. Generalized dirichlet distribution in bayesian analysis. *Applied Mathematics and Computation*, 97(2-3):165–181, 1998.

[131] Shasha Xie, Keelan Evanini, and Klaus Zechner. Exploring Content Features for Automated Speech Scoring. In *Proc. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2012.

[132] Helen Yannakoudakis. Automated assessment of English-learner writing. Technical Report UCAM-CL-TR-842, University of Cambridge Computer Laboratory, 2013.

[133] Su-Youn Yoon and Shasha Xie. Similarity-Based Non-Scorable Response Detection for Automated Speech Scoring. In *Proc. Ninth Workshop on Innovative Use of NLP for Building Educational Applications*, 2014.

[134] Su-Youn Yoon, Chong Min Lee, Ikkyu Choi, Xinhao Wang, Matthew Mulholland, and Keelan Evanini. Off-topic spoken response detection with word embeddings. In *INTERSPEECH*, pages 2754–2758, 2017.

[135] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. LSUN: construction of a large-scale image dataset using deep learning with humans in the loop, 2015. URL http://arxiv.org/abs/1506.03365. arXiv:1506.03365.

[136] Z. Yu, V. Ramanarayanan, D. Suendermann-Oeft, X. Wang, K. Zechner, L. Chen, J. Tao, A. Ivanou, and Y. Qian. Using bidirectional LSTM recurrent neural networks to learn high-level abstractions of sequential features for automated scoring of non-native spontaneous speech. In *Proc. Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 338–345, Dec 2015. doi: 10.1109/ASRU.2015.7404814.

[137] Klaus Zechner et al. Automatic scoring of non-native spontaneous speech in tests of spoken English. *Speech Communication*, 51(10):883–895, 2009.

[138] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[139] Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–663, 2018.

Appendix A

Derivations of Uncertainty Measures

The current appendix details the derivation of measures of uncertainty introduced in chapter 4. Section A.1 describes derivations of measures of uncertainty for Prior Networks which parameterize the Dirichlet distribution, while section A.2 describes derivations of measures of uncertainty for Prior Networks which parameterize the Normal-Inverse-Wishart distribution.

A.1 Dirichlet Prior Networks

The current section details the derivation of differential entropy, mutual information and expected pairwise KL-divergence for a Prior Network which parameterizes the Dirichlet distribution:

$$\begin{aligned} \mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}}) &= \mathtt{Dir}(\boldsymbol{\pi};\hat{\boldsymbol{\alpha}}) \\ \hat{\boldsymbol{\alpha}} &= \boldsymbol{f}(\boldsymbol{x}^*;\hat{\boldsymbol{\theta}}) \end{aligned} \tag{A.1}$$

where $\mathtt{p}(\boldsymbol{\pi};\hat{\boldsymbol{\alpha}})$ is a prior distribution over categorical distributions. The Dirichlet distribution is defined as:

$$\begin{aligned} \mathtt{Dir}(\boldsymbol{\pi};\boldsymbol{\alpha}) &= \mathcal{C}(\boldsymbol{\alpha})\prod_{c=1}^{K}\pi_c^{\alpha_c-1}, \quad \alpha_c > 0 \\ \mathcal{C}(\boldsymbol{\alpha}) &= \frac{\Gamma(\alpha_0)}{\prod_{c=1}^{K}\Gamma(\alpha_c)}, \quad \alpha_0 = \sum_{c=1}^{K}\alpha_c \end{aligned} \tag{A.2}$$

where $\Gamma(\cdot)$ is the *Gamma function*.

A.1.1 Differential Entropy

The differential entropy of the Dirichlet distribution can be derived as follows:

$$\begin{aligned}\mathcal{H}[\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})] =& -\mathbb{E}_{\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{x};\hat{\boldsymbol{\theta}})}[\ln(\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{x};\hat{\boldsymbol{\theta}}))] \\ =& \sum_{c=1}^{K} \ln\Gamma(\hat{\alpha}_c) - \ln\Gamma(\hat{\alpha}_0) - \sum_{c=1}^{K}(\hat{\alpha}_c - 1)\mathbb{E}_{\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{x};\hat{\boldsymbol{\theta}})}[\ln\pi_c] \\ =& \sum_{c=1}^{K} \ln\Gamma(\hat{\alpha}_c) - \ln\Gamma(\hat{\alpha}_0) - \sum_{c=1}^{K}(\hat{\alpha}_c - 1)\cdot\big(\psi(\hat{\alpha}_c) - \psi(\hat{\alpha}_0)\big)\end{aligned} \tag{A.3}$$

where ψ is the *digamma function* and $\mathbb{E}_{\mathrm{p}(\boldsymbol{\pi}|\hat{\boldsymbol{\alpha}})}[\ln(\pi_c)] = \psi(\hat{\alpha}_c) - \psi(\hat{\alpha}_0)$ is a standard result.

A.1.2 Mutual Information

The mutual information between the labels y and the categorical $\boldsymbol{\pi}$ for a Dirichlet distribution can be calculated as follows, using the fact that mutual information is the difference of the entropy of the expected distribution and the expected entropy of the distribution.

$$\begin{aligned}\underbrace{\mathcal{I}[y,\boldsymbol{\pi}|\boldsymbol{x}^*,\hat{\boldsymbol{\theta}}]}_{Knowledge\ Uncertainty} &= \underbrace{\mathcal{H}[\mathbb{E}_{\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{x}^*,\hat{\boldsymbol{\theta}})}[\mathrm{P}(y|\boldsymbol{\pi})]]}_{Total\ Uncertainty} - \underbrace{\mathbb{E}_{\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{x}^*,\hat{\boldsymbol{\theta}})}[\mathcal{H}[\mathrm{P}(y|\boldsymbol{\pi})]]}_{Expected\ Data\ Uncertainty} \\ &= \mathcal{H}[\mathrm{P}(y|\boldsymbol{x}^*,\hat{\boldsymbol{\theta}})] + \sum_{c=1}^{K}\mathbb{E}_{\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{x}^*,\hat{\boldsymbol{\theta}})}[\pi_c\ln\pi_c] \\ &= -\sum_{c=1}^{K}\frac{\hat{\alpha}_c}{\hat{\alpha}_0}\Big(\ln\frac{\hat{\alpha}_c}{\hat{\alpha}_0} - \psi(\hat{\alpha}_c+1) + \psi(\hat{\alpha}_0+1)\Big)\end{aligned} \tag{A.4}$$

The second term in this derivation is a non-standard result. The expected entropy of the distribution can be calculated in the following way:

$$\begin{aligned}\mathbb{E}_{\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{x}^*,\hat{\boldsymbol{\theta}})}[\pi_c\ln\pi_c] &= \frac{\Gamma(\hat{\alpha}_0)}{\prod_{c=1}^{K}\Gamma(\hat{\alpha}_c)}\int_{\mathcal{S}_K}\pi_c\ln\pi_c\prod_{c=1}^{K}\pi_c^{\hat{\alpha}_c-1}d\boldsymbol{\pi} \\ &= \frac{\hat{\alpha}_c}{\hat{\alpha}_0}\frac{\Gamma(\hat{\alpha}_0+1)}{\Gamma(\hat{\alpha}_c+1)\prod_{c'=1,\neq c}^{K}\Gamma(\hat{\alpha}_{c'})}\int_{\mathcal{S}_K}\pi_c^{\hat{\alpha}_c}\ln\pi_c\prod_{c'=1,\neq c}^{K}\pi_{c'}^{\hat{\alpha}_{c'}-1}d\boldsymbol{\pi} \\ &= \frac{\hat{\alpha}_c}{\hat{\alpha}_0}\big(\psi(\hat{\alpha}_c+1) - \psi(\hat{\alpha}_0+1)\big)\end{aligned} \tag{A.5}$$

Here the expectation is calculated by noting that the standard result of the expectation of $\ln\pi_c$ with respect to a Dirichlet distribution can be used if the extra factor π_c is accounted for

by adding 1 to the associated concentration parameter $\hat{\alpha}_c$ and multiplying by $\frac{\hat{\alpha}_c}{\hat{\alpha}_0}$ in order to have the correct normalizing constant.

A.1.3 Expected Pairwise KL-divergence

Similarly, the Expected Pairwise KL-divergence can also be analytically calculated for the Dirichlet distribution using the following derivation:

$$\begin{aligned}\mathcal{K}[\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})] &= \mathbb{E}_{\mathtt{p}(\boldsymbol{\pi}^{(1)}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}}),\mathtt{p}(\boldsymbol{\pi}^{(2)}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})}\big[\mathtt{KL}[\mathtt{P}(y|\boldsymbol{\pi}^{(1)})||\mathtt{P}(y|\boldsymbol{\pi}^{(2)})]\big] \\ &= -\sum_{c=1}^{K}\mathbb{E}_{\mathtt{p}(\boldsymbol{\pi}^{(1)}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})}[\mathtt{P}(\omega_c|\boldsymbol{\pi}^{(1)})]\mathbb{E}_{\mathtt{p}(\boldsymbol{\pi}^{(2)}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})}[\ln \mathtt{P}(\omega_c|\boldsymbol{\pi}^{(2)})] \\ &\quad - \mathbb{E}_{\mathtt{p}(\boldsymbol{\pi}^{(1)}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})}\big[\mathcal{H}[\mathtt{P}(y|\boldsymbol{\pi}^{(1)})]\big] \\ &= \sum_{c=1}^{K}\mathbb{E}_{\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})}[\pi_c \ln \pi_c] - \sum_{c=1}^{K}\mathbb{E}_{\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})}[\pi_c]\mathbb{E}_{\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})}[\ln \pi_c]\end{aligned} \tag{A.6}$$

The last step is valid only if $\mathtt{p}(\boldsymbol{\pi}^{(1)}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}}) = \mathtt{p}(\boldsymbol{\pi}^{(2)}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}}) = \mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})$, which represents independent draws of categorical from the Dirichlet. This expression then leads to a particularly elegant solution:

$$\begin{aligned}\mathcal{K}[\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}})] &= \sum_{c=1}^{K}\frac{\hat{\alpha}_c}{\hat{\alpha}_0}\big(\psi(\hat{\alpha}_c+1) - \psi(\hat{\alpha}_0+1)\big) - \sum_{c=1}^{K}\frac{\hat{\alpha}_c}{\hat{\alpha}_0}\big(\psi(\hat{\alpha}_c) - \psi(\hat{\alpha}_0)\big) \\ &= \frac{K-1}{\hat{\alpha}_0}\end{aligned} \tag{A.7}$$

Thus, the expected pairwise KL-divergence is inversely proportional to the concentration of the Dirichlet and is maximized when the concentration $\hat{\alpha}_0$ tends to 0.

A.2 Normal-inverse-Wishart Prior Networks

The current section details the derivation of differential entropy, mutual information and expected pairwise KL-divergence for regression Prior Networks which parameterizes the Normal-inverse Wishart distribution:

$$\begin{aligned}\mathtt{p}(\boldsymbol{\mu},\boldsymbol{\Sigma}|\boldsymbol{x}^*;\hat{\boldsymbol{\theta}}) &= \mathcal{NW}^{-1}(\boldsymbol{\mu},\boldsymbol{\Sigma};\hat{\boldsymbol{m}},\hat{\boldsymbol{S}},\hat{\kappa},\hat{\nu}) \\ \{\hat{\boldsymbol{m}},\hat{\boldsymbol{S}},\hat{\kappa},\hat{\nu}\} &= \boldsymbol{f}(\boldsymbol{x}^*;\hat{\boldsymbol{\theta}}),\ \hat{\kappa}>0,\ \hat{\nu}>K-1\end{aligned} \tag{A.8}$$

where $\boldsymbol{m}$ and $\boldsymbol{S}$ are the *prior mean* and the positive-definite *prior scatter matrix*, while κ and ν are the strengths of belief in each prior, respectively. The parameters κ and ν are conceptually similar to *precision* of the Dirichlet distribution α_0. The Normal-inverse-Wishart is a compound distribution which decomposes into a product of a conditional normal distribution over the mean and an inverse-Wishart distribution over the covariance:

$$\mathcal{NW}^{-1}(\boldsymbol{\mu}, \boldsymbol{\Sigma}; \boldsymbol{m}, \boldsymbol{S}, \kappa, \nu) = \mathcal{N}(\boldsymbol{\mu}; \boldsymbol{m}, \frac{1}{\kappa}\boldsymbol{\Sigma}) \cdot \mathcal{W}^{-1}(\boldsymbol{\Sigma}; \boldsymbol{S}, \nu) \tag{A.9}$$

The inverse-Wishart distribution $\mathcal{W}^{-1}$ is a distribution over positive-definite symmetric matrices $\boldsymbol{\Sigma}$ of size $K \times K$ defined as follows:

$$\mathcal{W}^{-1}(\boldsymbol{\Sigma}; \boldsymbol{S}, \nu) = \frac{|\boldsymbol{S}|^{\frac{\nu}{2}}}{2^{\frac{\nu K}{2}}\Gamma_K(\frac{\nu}{2})}|\boldsymbol{\Sigma}|^{\frac{-(\nu+K+1)}{2}} e^{-\frac{1}{2}\mathtt{tr}(\boldsymbol{S}\boldsymbol{\Sigma}^{-1})}, \ \nu \geq K-1 \tag{A.10}$$

where $\Gamma_K(\cdot)$ is the *multivariate gamma function* and K is the dimensionality of $\boldsymbol{y}$.

The derivation of measures of uncertainty in this appendix rely heavily on results from [46], specifically propositions 1-6, which are used to obtain closed for expression of expectations of logs and traces of covariance matrices.

A.2.1 Differential entropy of $\mathcal{NW}^{-1}$ Predictive Posterior

As discussed in section 4.3, the predictive posterior of a Prior Network which parameterizes a Normal-inverse-Wishart distribution is a multivariate students T distribution:

$$\mathbb{E}_{\mathtt{p}(\boldsymbol{\mu},\boldsymbol{\Sigma})}[\mathtt{p}(\boldsymbol{y}|\boldsymbol{\mu}, \boldsymbol{\Sigma})] = \mathtt{St}(\boldsymbol{y}|\boldsymbol{m}, \frac{\kappa+1}{\kappa(\nu-K+1)}\boldsymbol{S}, \nu-K+1) \tag{A.11}$$

The differential entropy of a standard multivariate student's T distribution with an identity scatter matrix $\boldsymbol{\Sigma} = \boldsymbol{I}$ is given by:

$$\mathcal{H}[\mathtt{St}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{I}, \nu)] = -\ln \frac{\Gamma(\frac{\nu+K}{2})}{\Gamma(\frac{\nu}{2})(\nu\pi)^{\frac{K}{2}}} + (\frac{\nu+K}{2}) \cdot \big(\psi(\frac{\nu+K}{2}) - \psi(\frac{\nu}{2})\big) \tag{A.12}$$

which is a result obtained from [7]. Using the property of differential entropy [23], that if $\boldsymbol{x} \sim \mathtt{p}(\boldsymbol{x})$ and $\boldsymbol{y} = \boldsymbol{\mu} + \boldsymbol{A}\boldsymbol{x}$, then:

$$\mathcal{H}[\mathtt{p}(\boldsymbol{y})] = \mathcal{H}[\mathtt{p}(\boldsymbol{x})] + \ln|\boldsymbol{A}| \tag{A.13}$$

we can show that the differential entropy of a standard general multivariate student's T distribution is given by:

$$\mathcal{H}[\mathtt{St}(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma},\nu)] = \frac{1}{2}\ln|\boldsymbol{\Sigma}| - \ln\frac{\Gamma(\frac{\nu+K}{2})}{\Gamma(\frac{\nu}{2})(\nu\pi)^{\frac{K}{2}}} + (\frac{\nu+K}{2})\cdot\big(\psi(\frac{\nu+K}{2}) - \psi(\frac{\nu}{2})\big) \quad \text{(A.14)}$$

Using this expression, we can show that the differential entropy of the predictive posterior of a Normal-inverse-Wishart Prior Network is given by:

$$\begin{aligned}\mathcal{H}\big[\mathbb{E}_{\mathtt{p}(\boldsymbol{\mu},\boldsymbol{\Sigma})}[\mathtt{p}(\boldsymbol{y}|\boldsymbol{\mu},\boldsymbol{\Sigma})]\big] &= \mathcal{H}\big[\mathtt{St}(\boldsymbol{y}|\boldsymbol{m},\frac{\kappa+1}{\kappa(\nu-K+1)}\boldsymbol{S},\nu-K+1)\big] \\ &= -\ln\frac{\Gamma(\frac{\nu+1}{2})}{\Gamma(\frac{\nu-K+1}{2})\big((\nu-K+1)\pi\big)^{\frac{K}{2}}} \\ &\quad + (\frac{\nu+1}{2})\big(\psi(\frac{\nu+1}{2}) - \psi(\frac{\nu-K+1}{2})\big) \\ &\quad + \frac{1}{2}\ln|\boldsymbol{S}| + \frac{K}{2}\ln\frac{\kappa+1}{\kappa(\nu-K+1)}\end{aligned} \quad \text{(A.15)}$$

A.2.2 Differential entropy of Normal-inverse-Wishart distribution

The differential entropy of the Normal-inverse-Wishart distribution can be derived by separarely considering the differential entopy of the inverse-Wishart distribution $\mathcal{W}^{-1}(\boldsymbol{\Sigma})$ and the expected differential entropy of the Normal distribution $\mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\Sigma})$:

$$\mathcal{H}[\mathcal{N}\mathcal{W}^{-1}(\boldsymbol{\mu},\boldsymbol{\Sigma})] = \mathbb{E}_{\mathcal{W}^{-1}(\boldsymbol{\Sigma})}\big[\mathcal{H}[\mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\Sigma})]\big] + \mathcal{H}\big[\mathcal{W}^{-1}(\boldsymbol{\Sigma})\big] \quad \text{(A.16)}$$

The result for the differential entropy of the inverse-Wishart distribution is quoted from [46]:

$$\begin{aligned}\mathcal{H}\big[\mathcal{W}^{-1}(\boldsymbol{\Sigma})\big] &= \ln\Gamma_K(\frac{\nu}{2}) + \frac{\nu K}{2} + \frac{K+1}{2}\ln\Big|\frac{\boldsymbol{S}}{2}\Big| \\ &\quad - \frac{\nu+K+1}{2}\sum_{c=1}^{K}\psi\big(\frac{\nu-K+c}{2}\big)\end{aligned} \quad \text{(A.17)}$$

The expected differential entropy of $\mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\Sigma})$ is obtained by applying proposition 3 from [46]:

$$\begin{aligned}\mathbb{E}_{\mathcal{W}^{-1}(\boldsymbol{\Sigma})}\big[\mathcal{H}[\mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\Sigma})]\big] &= \mathbb{E}_{\mathcal{W}^{-1}(\boldsymbol{\Sigma})}\Big[\mathcal{H}\big[\mathcal{N}\big(\boldsymbol{\mu}|\boldsymbol{m}, \boldsymbol{\Sigma}\frac{1}{\kappa}\big)\big]\Big] \\ &= \frac{1}{2}\mathbb{E}_{\mathcal{W}^{-1}(\boldsymbol{\Sigma})}\Big[\ln|\boldsymbol{\Sigma}| + K + K\ln\frac{2\pi}{\kappa}\Big] \\ &= \frac{1}{2}\Big(\ln|\frac{\boldsymbol{S}}{2}| - \sum_{c=1}^{K}\psi\big(\frac{\nu - K + c}{2}\big) + K + K\ln\frac{2\pi}{\kappa}\Big)\end{aligned} \tag{A.18}$$

Thus, the differential entropy of the Normal-inverse-Wishart distribution is given by the following expression:

$$\begin{aligned}\mathcal{H}[\mathcal{N}\mathcal{W}^{-1}(\boldsymbol{\mu}, \boldsymbol{\Sigma})] = &\ln\Gamma_K(\frac{\nu}{2}) + \frac{(\nu+1)K}{2} + \frac{K+2}{2}\ln|\frac{\boldsymbol{S}}{2}| \\ &- \frac{\nu + K + 2}{2}\sum_{c=1}^{K}\psi\big(\frac{\nu - K + c}{2}\big) + K\ln\frac{\pi}{\kappa}\end{aligned} \tag{A.19}$$

A.2.3 Mutual Information Derivations

The mutual information between the target $\boldsymbol{y}$ and the parameters of the output distribution $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$, the prior of which is given by the Normal-inverse-Wishart distribution can be obtained by considering that the mutual information is the sum of the differential entropy of the predictive posterior and the expected differential entropy of the output:

$$\mathcal{I}[\boldsymbol{y}, \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}] = \mathcal{H}\big[\mathbb{E}_{\mathcal{N}\mathcal{W}^{-1}(\boldsymbol{\mu},\boldsymbol{\Sigma})}[\mathrm{p}(\boldsymbol{y}|\boldsymbol{\mu}, \boldsymbol{\Sigma})]\big] - \mathbb{E}_{\mathcal{N}\mathcal{W}^{-1}(\boldsymbol{\mu},\boldsymbol{\Sigma})}\big[\mathcal{H}[\mathrm{p}(\boldsymbol{y}|\boldsymbol{\mu}, \boldsymbol{\Sigma})]\big] \tag{A.20}$$

The closed form expression for the first term was previously derived and is given by equation A.15. The closed form solution for the second term is obtained as follows by applying proposition 3 from [46]:

$$\begin{aligned}\mathbb{E}_{\mathcal{N}\mathcal{W}^{-1}(\boldsymbol{\mu},\boldsymbol{\Sigma})}[\mathcal{H}[\mathrm{p}(\boldsymbol{y}|\boldsymbol{\mu}, \boldsymbol{\Sigma})]] &= \frac{1}{2}\mathbb{E}_{\mathcal{W}^{-1}(\boldsymbol{\Sigma})}\Big[K + K\ln 2\pi + \ln|\boldsymbol{\Sigma}|\Big] \\ &= \frac{1}{2}\Big(K + K\ln 2\pi + \ln\Big|\frac{\boldsymbol{S}}{2}\Big| - \sum_{c=1}^{K}\psi\big(\frac{\nu - K + c}{2}\big)\Big)\end{aligned} \tag{A.21}$$

Thus, the closed form expression for the mutual information is:

$$\begin{aligned}\mathcal{I}[\boldsymbol{y}, \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}] = & \ln\Gamma\big(\frac{\nu-K+1}{2}\big) - \ln\Gamma\big(\frac{\nu+1}{2}\big) \\ & + \ (\frac{\nu+1}{2})\Big(\psi(\frac{\nu+1}{2}) - \psi(\frac{\nu-K+1}{2})\Big) \\ & + \ \frac{K}{2}\Big(\ln\big(\frac{\kappa+1}{\kappa}\big) - 1\Big) + \frac{1}{2}\sum_{c=1}^{K}\psi(\frac{\nu-K+c}{2})\end{aligned} \tag{A.22}$$

A.2.4 Expected Pairwise KL-divergence

Finally, expected pairwise KL-divergence between independently drawn multivariate normal distributions drawn from a Norma-inverse-Wishart Prior is provided below, using proposition 3, 5 and 6 from [46]:

$$\begin{aligned}\mathcal{K}\big[\mathcal{NW}^{-1}(\boldsymbol{\mu}, \boldsymbol{\Sigma})\big] &= \mathbb{E}_{\mathtt{p}(\boldsymbol{\mu}_0,\boldsymbol{\Sigma}_0)\mathtt{p}(\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1)}\Big[\mathtt{KL}\big[\mathtt{p}(\boldsymbol{y}|\boldsymbol{\mu}_0,\boldsymbol{\Sigma}_0)||\mathtt{p}(\boldsymbol{y}|\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1)\big]\Big] \\ &= \frac{1}{2}\mathbb{E}_{\mathtt{p}(\boldsymbol{\mu}_0,\boldsymbol{\Sigma}_0)\mathtt{p}(\boldsymbol{\mu}_1,\boldsymbol{\Sigma}_1)}\Big[\mathtt{tr}(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0) - K + \ln|\boldsymbol{\Sigma}_1| \\ &- \ \ln|\boldsymbol{\Sigma}_0|\mathtt{tr}(\Sigma_1^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)^{\mathtt{T}})\Big] \\ &= \frac{1}{2}\Big(\frac{\nu K}{\nu-K-1} - K\Big) + \frac{1}{2\kappa}\Big(\frac{\nu K}{\nu-K-1} + K\Big)\end{aligned} \tag{A.23}$$

where it is necessary to point out that $\mathtt{p}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) = \mathtt{p}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) = \mathcal{NW}^{-1}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. This is an elegant solution, similarly to the derivation for the expected pairwise KL-divergence between independent drawn from the Dirichlet distribution derived in the previous section.

Appendix B

Symmetries in Forward and Reverse KL-divergences

In this appendix we explore symmetric properties of KL-divergences between conjugate priors, where forward KL-divergence is defined as $\texttt{KL}[p||q]$ and reverse KL-divergence is $\texttt{KL}[q||p]$, where p is the true prior distribution and q is the model. In this appendix it is shown that a forward KL-divergence between conjugate priors implies a reverse cross-entropy between distributions which the priors are over, while a reverse KL-divergence between conjugate priors implies a forward cross entropy between distributions which the priors are over. These properties are demonstrated for the Dirichlet and Normal-inverse-Wishart distributions. It is not known to the author whether the described property is known as a general property of α-divergences between members of the exponential family of distributions.

B.1 Dirichlet Distribution

In this section we analyze properties of forward and reverse KL-divergences between Dirichlet distributions, which is a conjugate prior to the categorical distribution and a member of the exponential family of distributions.

The Dirichlet distribution is defined as:

$$\begin{aligned} \texttt{p}(\boldsymbol{\pi};\boldsymbol{\alpha}) &= \mathcal{C}(\boldsymbol{\alpha})\prod_{c=1}^{K}\pi_c^{\alpha_c-1}, \quad \alpha_c > 0 \\ \mathcal{C}(\boldsymbol{\alpha}) &= \frac{\Gamma(\alpha_0)}{\prod_{c=1}^{K}\Gamma(\alpha_c)}, \quad \alpha_0 = \sum_{c=1}^{K}\alpha_c \end{aligned} \tag{B.1}$$

The mean and mode of the Dirichlet are given by:

$$\begin{aligned} \text{Mean} \rightarrow \hat{\pi}_c &= \frac{\alpha_c}{\alpha_0} \\ \text{Mode} \rightarrow \tilde{\pi}_c &= \frac{\alpha_c - 1}{\alpha_0 - K} \end{aligned} \tag{B.2}$$

where Γ is the gamma function. The KL-divergence between a target Dirichlet $\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{\beta})$ and an approximating Dirichlet $\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{\alpha})$ can be decomposed into the negative differential entropy of the target and the cross entropy between the target and the approximation:

$$\mathtt{KL}[\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{\beta})||\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{\alpha})] = \underbrace{-\mathbb{E}_{\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{\beta})}\big[\ln \mathtt{p}(\boldsymbol{\pi}|\boldsymbol{\alpha})\big]}_{Cross-Entropy} - \underbrace{\mathcal{H}\big[\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{\beta})\big]}_{Diff.\ Entropy} \tag{B.3}$$

Let's consider only the cross-entropy term and obtain a closed-form expression for it:

$$\begin{aligned} \mathcal{L}^{CE}(\boldsymbol{\beta}, \boldsymbol{\alpha}) &= -\ln \mathcal{C}(\boldsymbol{\alpha}) - \sum_{c=1}^{K} (\alpha_c - 1)\mathbb{E}_{\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{\beta})}[\ln \pi_c] \\ &= -\ln \mathcal{C}(\boldsymbol{\alpha}) - (\alpha_0 - K) \sum_{c=1}^{K} \frac{\alpha_c - 1}{\alpha_0 - K} \cdot \big(\psi(\beta_c) - \psi(\beta_0)\big) \end{aligned} \tag{B.4}$$

where ψ is the digamma function. We can consider the following series approximation to the digamma function:

$$\begin{aligned} \psi(x) &= \ln x - \frac{1}{2x} + \mathcal{O}(x^2) \\ &\approx \ln x - \frac{1}{2x} \end{aligned} \tag{B.5}$$

Using this approximation, we can show that the cross entropy between Dirichlet distributions contains within it the *reverse* cross entropy between the mode of $\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{\alpha})$ and the mean of $\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{\beta})$:

$$\begin{aligned} \mathcal{L}^{CE}(\boldsymbol{\beta}, \boldsymbol{\alpha}) &= \sum_{c=1}^{K} (\alpha_c - 1) \cdot \big(\frac{\beta_0 - \beta_c}{2\beta_c \beta_0}\big) - \ln \mathcal{C}(\boldsymbol{\alpha}) \\ &\quad - (\alpha_0 - K) \sum_{c=1}^{K} \frac{\alpha_c - 1}{\alpha_0 - K} \cdot \ln \frac{\beta_c}{\beta_0} \\ &= \sum_{c=1}^{K} (\alpha_c - 1) \cdot \big(\frac{\beta_0 - \beta_c}{2\beta_c \beta_0}\big) - \ln \mathcal{C}(\boldsymbol{\alpha}) \\ &\quad + (\alpha_0 - K) \cdot \mathcal{L}^{CE}(\tilde{\boldsymbol{\pi}}^{(\alpha)}, \hat{\boldsymbol{\pi}}^{(\beta)}) \end{aligned} \tag{B.6}$$

Similarly, we can show that the *reverse* entropy between Dirichlet distributions contains within it the *forward* cross entropy between the mode of $\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{\beta})$ and the mean of $\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{\alpha})$:

$$\begin{aligned}\mathcal{L}^{CE}(\boldsymbol{\alpha},\boldsymbol{\beta}) = &\sum_{c=1}(\beta_c - 1)\cdot\big(\frac{\alpha_0-\alpha_c}{2\alpha_c\alpha_0}\big) - \ln\mathcal{C}(\boldsymbol{\beta}) \\ &- (\beta_0 - K)\cdot\mathcal{L}^{CE}(\tilde{\boldsymbol{\pi}}^{(\beta)},\hat{\boldsymbol{\pi}}^{(\alpha)})\end{aligned} \tag{B.7}$$

This is a curious anti-symmetric property that the *forward* KL-divergence between conjugate priors to a distribution implies a *reverse* cross entropy between those distributions, and vice-versa.

B.2 Normal-inverse-Wishart Distribution

Having explored the properties of forward and reverse KL-divergences between Dirichlet distribution, we now explore the same for Normal-inverse-Wishart distributions. The Normal-inverse-Wishart is a compound distribution which decomposes into a product of a conditional normal distribution over the mean and an inverse-Wishart distribution over the covariance:

$$\begin{aligned}\mathcal{NW}^{-1}(\boldsymbol{\mu},\boldsymbol{\Sigma};\boldsymbol{m},\boldsymbol{S},\kappa,\nu) &= \mathcal{N}(\boldsymbol{\mu};\boldsymbol{m},\frac{1}{\kappa}\boldsymbol{\Sigma})\cdot\mathcal{W}^{-1}(\boldsymbol{\Sigma};\boldsymbol{S},\nu) \\ \boldsymbol{\theta} &= \{\boldsymbol{m},\boldsymbol{S},\kappa,\nu\}\end{aligned} \tag{B.8}$$

The cross-entropy between a target normal distribution $(N)_1$ and a model $\mathcal{N}_0$ is:

$$\mathcal{L}^{CE}(\mathcal{N}_1,\mathcal{N}_0) = \frac{1}{2}\Big[\mathtt{tr}\Big(\boldsymbol{\Sigma}_0^{-1}\big(\boldsymbol{\Sigma}_1 + (\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)^{\mathtt{T}}\big) + K\ln 2\pi e + \ln|\boldsymbol{\Sigma}_0|\Big] \tag{B.9}$$

while the reverse cross-entropy is given by:

$$\mathcal{L}^{CE}(\mathcal{N}_0,\mathcal{N}_1) = \frac{1}{2}\Big[\mathtt{tr}\Big(\boldsymbol{\Sigma}_1^{-1}\big(\boldsymbol{\Sigma}_0 + (\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)^{\mathtt{T}}\big) + K\ln 2\pi e + \ln|\boldsymbol{\Sigma}_1|\Big] \tag{B.10}$$

The KL-divergence between Normal-inverse-Wishart distribution can be decomposed as follows:

$$\mathcal{L}^{KL}(\mathcal{NW}_1^{-1},\mathcal{NW}_0^{-1}) = \underbrace{-\mathbb{E}_{\mathrm{p}(\boldsymbol{\mu},\boldsymbol{\Sigma};\boldsymbol{\theta}_1)}[\ln\mathrm{p}(\boldsymbol{\mu},\boldsymbol{\Sigma};\boldsymbol{\theta}_0)]}_{Cross\ Entropy} - \underbrace{\mathcal{H}[\mathrm{p}(\boldsymbol{\mu},\boldsymbol{\Sigma};\boldsymbol{\theta}_1)]}_{Differential\ Entropy} \tag{B.11}$$

As before, let's focus on the cross-entropy term:

$$\begin{aligned}
\mathcal{L}^{CE}(\mathcal{NW}_0^{-1}, \mathcal{NW}_1^{-1}) = & -\mathbb{E}_{\mathtt{p}(\boldsymbol{\mu},\boldsymbol{\Sigma}|\boldsymbol{\theta}_1)}\big[\ln \mathtt{p}(\boldsymbol{\mu}, \boldsymbol{\Sigma}; \boldsymbol{\theta}_0)\big] \\
= & \,\mathbb{E}_{\mathtt{p}(\boldsymbol{\mu},\boldsymbol{\Sigma};\boldsymbol{\theta}_1)}\Big[\frac{(\nu_0 + K + 2)}{2}\ln|\boldsymbol{\Sigma}| \\
& + \frac{1}{2}\mathtt{tr}\big(\boldsymbol{\Sigma}^{-1}(\boldsymbol{S}_0 + \kappa_0(\boldsymbol{\mu} - \boldsymbol{m}_0)(\boldsymbol{\mu} - \boldsymbol{m}_0)^{\mathtt{T}})\big)\Big] \\
& + \frac{1}{2}\Big(K\ln 2\pi + \nu_0 K \ln 2 + 2\ln\Gamma_K(\frac{\nu_0}{2}) - \nu_0 \ln|\boldsymbol{S}_0|\Big) \\
= & \,\frac{(\nu_0 + K + 2)}{2}\Big(\ln|\boldsymbol{S}_1| - \sum_{c=1}^{K}\psi\big(\frac{\nu_0 - K + c}{2}\big)\Big) + \frac{K\kappa_0}{2\kappa_1} \\
& + \frac{1}{2}\mathtt{tr}\big(\nu_1\boldsymbol{S}_1^{-1}(\boldsymbol{S}_0 + \kappa_0(\boldsymbol{m}_1 - \boldsymbol{m}_0)(\boldsymbol{m}_1 - \boldsymbol{m}_0)^{\mathtt{T}})\big)\Big] \\
& + \frac{1}{2}\Big(K\ln 2\pi + \nu_0 K \ln 2 + 2\ln\Gamma_K(\frac{\nu_0}{2}) - \nu_0 \ln|\boldsymbol{S}_0|\Big)
\end{aligned} \tag{B.12}$$

We an see that this form is similar to the reverse cross-entropy between normal distributions, presented in equation B.10. At the same time, the expression for the reverse KL-divergence between Normal-inverse-Wishart distributions, provided below, is similar to the form of expression for the forward cross entropy between normal distributions, provdided in equation B.9.

$$\begin{aligned}
\mathcal{L}^{CE}(\mathcal{NW}_1^{-1}, \mathcal{NW}_0^{-1}) = & \,\frac{(\nu_1 + K + 2)}{2}\Big(\ln|\boldsymbol{S}_0| - \sum_{c=1}^{K}\psi\big(\frac{\nu_1 - K + c}{2}\big)\Big) + \frac{K\kappa_1}{2\kappa_0} \\
& + \frac{1}{2}\mathtt{tr}\big(\nu_0\boldsymbol{S}_0^{-1}(\boldsymbol{S}_1 + \kappa_1(\boldsymbol{m}_1 - \boldsymbol{m}_0)(\boldsymbol{m}_1 - \boldsymbol{m}_0)^{\mathtt{T}})\big)\Big] \\
& + \frac{1}{2}\Big(K\ln 2\pi + \nu_1 K \ln 2 + 2\ln\Gamma_K(\frac{\nu_1}{2}) - \nu_1 \ln|\boldsymbol{S}_0|\Big)
\end{aligned} \tag{B.13}$$

Appendix C

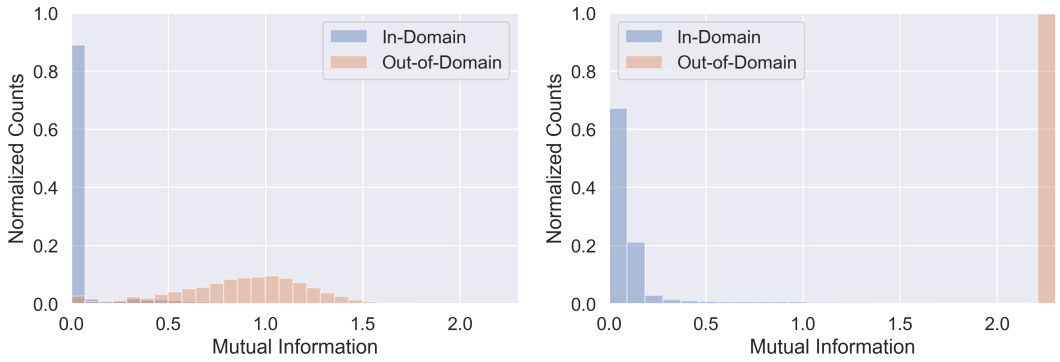
SVHN Out-of-Domain Detection

In section 5.4 is was established that Prior Networks outperform the baseline and display the desired properties on the MNIST and CIFAR-10 datasets. Here we consider out-of-distribution input detection for models trained on the SVHN dataset. In the following experiments, the 10,000 images from the SVHN test set are used as in-domain data and the test sets of the CIFAR-100, LSUN and TinyImageNet datasets are out as out-of-distribution datasets. In all experiments there are equal amount of out-of-distribution and in-domain images.

OOD Data	Model	Total Uncertainty		Knowledge Uncertainty		
		Max.P	Ent.	M.I.	EPKL	D.Ent.
CIFAR100	DNN	91.7 $\pm$1.1	93.1 $\pm$0.7	-	-	-
	MCDP	92.5 $\pm$0.8	92.8 $\pm$0.9	92.2 $\pm$0.9	92.4 $\pm$0.9	-
	ENSM	97.1 $\pm$ NA	97.6 $\pm$ NA	97.7 $\pm$ NA	97.6 $\pm$ NA	-
	PN-RKL	99.7 $\pm$0.1	**99.8** $\pm$0.1	**99.8** $\pm$0.1	**99.8** $\pm$0.1	**99.8** $\pm$0.1
LSUN	DNN	88.6 $\pm$2.3	90.9 $\pm$1.8	-	-	-
	MCDP	90.2 $\pm$1.8	90.4 $\pm$1.8	90.0 $\pm$1.9	90.4 $\pm$1.8	-
	ENSM	97.7 $\pm$ NA	98.1 $\pm$ NA	98.5 $\pm$ NA	98.7 $\pm$ NA	-
	PN-RKL	**100.0** $\pm$0.0	**100.0** $\pm$0.0	**100.0** $\pm$0.0	**100.0** $\pm$0.0	**100.0** $\pm$0.0
TIM	DNN	90.7 $\pm$1.6	92.4 $\pm$1.3	-	-	-
	MCDP	91.8 $\pm$1.3	92.1 $\pm$1.4	91.7 $\pm$1.4	92.0 $\pm$1.2	-
	ENSM	97.7 $\pm$ NA	98.1 $\pm$ NA	98.4 $\pm$ NA	98.5 $\pm$ NA	-
	PN-RKL	**100.0** $\pm$0.0	**100.0** $\pm$0.0	**100.0** $\pm$0.0	**100.0** $\pm$0.0	**100.0** $\pm$0.0

Table C.1 SVHN out-of-domain detection results in terms of mean % AUROC $\pm 2\sigma$ across 10 models. Only a single set of results is obtained using explicit ensemble ENSM.

The results shown in table C.1 shows that in all experiments Prior Networks consistently achieves highest performance and are able to separate out-of-distribution images from the in-domain images. The second best model, is an explicit ensemble of DNNs, with Monte-Carlo Dropout ensembles and single DNNs yielding the worst performance. As SVHN is, overall, a low data uncertainty dataset, all measures of uncertainty yield similar performance, as expected. The histogram of uncertainty shown in figure C.1 show that the uncertainty which Prior Networks assign out-of-distribution data is far higher than for in-domain data, unlike the other models, which are yield low estimates of uncertainty for certain in-domain images. These results show that Prior Networks, given appropriate out-of-distribution training data and out-of-distribution test data which does not overlap with the in-domain region, are capable of perfectly separating out in-domain and out-of-distribution images.



(a) Explicit Ensemble (b) Prior Network

Figure C.1 Histogram of mutual information for in-domain (SVHN test set) and out-of-domain (TinyImageNet test set) images derived from explicit ensemble (ENSM) and Prior Network (PN-RKL). Predictions of 10 PN-RKL models trained from different random initializations are concatenated together.