MACHINE LEARNING FOR INFORMED REPRESENTATION LEARNING

Inauguraldissertation

zur
Erlangung der Würde eines Doktors der Philosophie
vorgelegt der
Philosophisch-Naturwissenschaftlichen Fakultät
der Universität Basel

von

Maxim Samarin

2022

Original document stored on the publication server of the University of Basel edoc.unibas.ch.
This work is licensed under the Creative Commons "Attribution – Non-Commercial – No Derivatives – 4.0 International" license (CC BY-NC-ND 4.0).

Genehmigt von der Philosophisch-Naturwissenschaftlichen Fakultät
auf Antrag von

Prof. Dr. Volker Roth, Dissertationsleiter

Prof. Dr. Heiko Schuldt, Zweitbetreuer

Prof. Dr. Mikhail Kanevski, Externer Gutachter

Basel, den 15. November 2022

Prof. Dr. Marcel Mayor, Dekan

Maxim Samarin: *Machine Learning for Informed Representation Learning*,
© 2022 (CC BY-NC-ND 4.0 license)

Für meine Familie.

We are drowning in information but starved for knowledge.

— John Naisbitt

ABSTRACT

The way we view reality and reason about the processes surrounding us is intimately connected to our perception and the representations we form about our observations and experiences. The popularity of machine learning and deep learning techniques in that regard stems from their ability to form useful representations by learning from large sets of observations. Typical application examples include image recognition or language processing for which artificial neural networks are powerful tools to extract regularity patterns or relevant statistics. In this thesis, we leverage and further develop this representation learning capability to address relevant but challenging real-world problems in geoscience and chemistry, to learn representations in an informed manner relevant to the task at hand, and reason about representation learning in neural networks, in general.

Firstly, we develop an approach for efficient and scalable semantic segmentation of degraded soil in alpine grasslands in remotely-sensed images based on convolutional neural networks. To this end, we consider different grassland erosion phenomena in several Swiss valleys. We find that we are able to monitor soil degradation consistent with state-of-the-art methods in geoscience and can improve detection of affected areas. Furthermore, our approach provides a scalable method for large-scale analysis which is infeasible with established methods.

Secondly, we address the question of how to identify suitable latent representations to enable generation of novel objects with selected properties. For this, we introduce a new deep generative model in the context of manifold learning and disentanglement. Our model improves targeted generation of novel objects by making use of property cycle consistency in property-relevant and property-invariant latent subspaces. We demonstrate the improvements on the generation of molecules with desired physical or chemical properties. Furthermore, we show that our model facilitates interpretability and exploration of the latent representation.

Thirdly, in the context of recent advances in deep learning theory and the neural tangent kernel, we empirically investigate the learning of feature representations in standard convolutional neural networks and corresponding random feature models given by the linearisation of the neural networks. We find that performance differences between standard and linearised networks generally increase with the difficulty of the task but decrease with the considered width or over-parametrisation of these networks. Our results indicate interesting implications for feature learning and random feature models as well as the generalisation performance of highly over-parametrised neural networks.

In summary, we employ and study feature learning in neural networks and review how we may use informed representation learning for challenging tasks.

ACKNOWLEDGEMENTS

I believe that every researcher at some point in their life feels a deep reverence for the accomplishments of bright minds that came before them, influential peers or the beauty of their science or field. Presumably, most of us can relate to that *if we have seen further than others, it is by standing on the shoulders of giants*. This popular metaphor of humility and continuing intellectual progress was once expressed by Isaac Newton, but is an idea which was formulated long before him and will surely persist when thinking about one's own contribution to science and knowledge. There are many people I have never met which allowed me to pursue my dissertation. But there are also several bright minds which have accompanied me along my way and to whom I would like to express my gratitude.

When I finished my physics degree, I had the sensation that there were some lose ends I would have liked to connect. Pursuing a Ph.D. in machine learning put me a long way forward on this path. And for this I deeply appreciate the opportunity my supervisor Volker Roth bestowed on me. In the last years, I have learned a lot from our discussions and critical reflections and left our meetings more knowledgeable than I entered them. I would like to thank Volker for the freedom he granted me and the support he offered me.

It was a pleasure to discuss my research with my second supervisor Heiko Schuldt. Heiko always provided encouragement and his considerations offered me different perspectives on my work and machine learning in general. I very much enjoyed collaborating with Heiko and his group in our project.

I am very grateful for the interest Mikhail Kanevski put into my work and for agreeing to serve as the external expert. I met Mikhail at the general assembly of the European Geoscience Union in 2018 for the first time, where he organised a track on machine learning. And with every next encounter, I looked forward to our conversations and the great variety of advice and suggestions he shared with me.

It was a particular pleasure to work with Lauren Zweifel on our project. Without fail, I could rely on her competence and assistance. Our trip to the Urseren valley is one of the memorable highlights of my doctorate. And

also my thanks goes to Christine Alewell for a great project collaboration and her encouragement of our work.

I am very happy to have joined the seminar on deep learning theory organised by David Belius which led to our project. I enjoyed the close collaboration with David and learned a lot from him.

A particular thanks goes to Ivan Dokmanić not only for chairing my defence but also for great discussions and offering his advice, and also to his group for intriguing lab meetings and a great time in the office.

I appreciate the support of the Swiss National Science Foundation for my project and I am happy to have been part of the National Research Programme 75 on big data.

Furthermore, I would like to express my gratitude towards the center for scientific computing, sciCORE, whose resources and exceptional support enabled my research to a great extent. But also to our IT team, Alexander Ugolini, Bas Kin, and Robert Frank, as well as our administrative team Patricia Krattiger, Yvonne Walser, and Heike Freiberger a big thanks!

When I started my Ph.D., I had great colleagues from which I learned so much and with whom I share so many valuable memories. Thank you very much for that Mario Wieser, Sebastian Keller, Damian Murezzan, Sonali Parbhoo, and Aleksander Wieczorek. And with time the circle of great colleagues grew to Fabricio Arend Torres, Monika Nagy-Huber, Vitali Nesterov, Daniel Hauke, and Marcello Negri to all of whom I want to say thank you for a great time!

I would like to thank the students who worked with me in various projects. I particularly want to thank Stephan Unter, Manvi Bhatia, and Benjamin Kessler.

Apart from our own group, it was always a pleasure to discuss research and enjoy a great time together with Thomas Vetter and his group. In particular, I would like to thank Adam Kortylewski, Dana Rahbani, Dennis Madsen, and Patrick Kahr. It was always great to have you around. And also thank you to Imant Daunhawer and Thomas Sutter for memorable times together!

I am in debt to Marcello Negri, Fabricio Arend Torres, Lauren Zweifel, Imant Daunhawer, Mario Wieser, Alexandra Moniz, and Jörg Wieder for reading parts of this thesis and providing invaluable feedback.

Early on in my studies, I was lucky to meet a few very special people in my life. And it is funny how all of us started with physics (with two exceptions) and how all of us went in quite different directions. And although we went different ways, we still went together. Thank you so much for that Christina Sauter, Elisa Bäumer, Rebecca Westphal, Marie Andrä, Matthäus Geiger, Maximilian Schwetz, and Jörg Wieder.

And a particular heartfelt thanks to Alex and Jörg for their friendship and support. You do not know how much it meant to me.

Finally, it is my family who helped me to reach where I am now. It is difficult to express my gratitude for your love and support and making me who I am. All of this is because of you.

CONTENTS

1 INTRODUCTION

The ability to draw conclusions from a plethora of information is a powerful advantage not only for individuals or organisations, but also for a society as a whole. And the ramifications of this ability can be profound. Acquiring large bodies of detailed observations of complex phenomena can assist in developing an understanding of their inner mechanisms and unravelling important correlations or causations. A prominent example of this is personalised medicine, where analysing extensive electronic health records of individuals is expected to revolutionise medicine by assisting medical experts in improving identification of disease characteristics and providing tailored therapies for patients (Zhang et al., 2019; Wilkinson et al., 2020). From a practitioner's point of view, it appears sensible to make use of these rich sources of knowledge in a great variety of applications. Yet collecting intimate information of individuals like health records or, more generally, other personal data on preferences or behavioural patterns might allow for unprecedented intrusions (Liu et al., 2021). In many different domains, these kinds of insights may be used to exert some sort of influence over individuals by rewarding desirable behaviour and penalising undesirable actions. Therefore, it is of tremendous societal relevance to reach a consensus on how we can benefit from the abundance of data while having regulatory frameworks on the usage in place. Furthermore, we need to find answers to who maintains control over the data, and where limitations of this powerful inference ability should lie. While these discussions are already ongoing and the benefits appear within reach, it still remains challenging to make use of *big data* and live up to the promises of – what is very broadly referred to as – *artificial intelligence*[1].

1.1 MACHINE LEARNING FOR CHALLENGING PROBLEMS

In order to move forward in this endeavour, it might be instrumental to ask: How do we generally approach a problem in statistics and machine learning? In his influential paper on *the two cultures of statistical modelling*

1 Deep learning $\subsetneq$ machine learning $\subsetneq$ artificial intelligence.

(Breiman, 2001), Leo Breiman describes two different approaches in statistics which he calls the *data modelling* and *algorithmic modelling culture*. Both aim to identify a functional relationship of inputs or predictors x with outputs or responses y for which a true unknown function, given by nature, is assumed to exist.

In the data modelling culture, the unknown function is approximated by a (stochastic) parametric model for inference and the quality of this approximation is assessed. This corresponds to the more traditional view on statistics and is illustrated by the solid path in Figure 1.1. A simple example is linear regression where response y is regressed on predictor x. Having this explicit model, devised by an expert, permits examining properties of the model and predicting responses y' to novel inputs x'. This is a typical approach pursued in the natural sciences which enables us to understand complex phenomena in nature. In physics, for instance, differential equations are used to describe the relation between predictor and response variables.

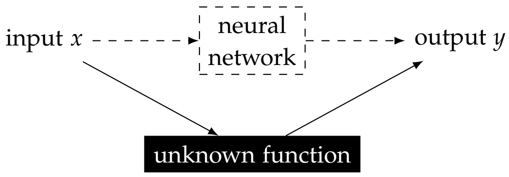


Figure 1.1: Illustration of the *data modelling* and *algorithmic modelling culture* (Breiman, 2001). We attempt to design a model which maps predictors x to responses y, either by (i) an explicit parametric model of the underlying unknown function or (ii) a black box algorithmic model like a neural network with no explicit model for the unknown function but which provides predictions y' for novel inputs x'.

On the other side, algorithmic modelling considers the unknown function to be potentially too complex for us to formalise appropriately. Thus, we attempt to identify an algorithmic approach to obtain a model which does not explicitly grant us any insight into the unknown function, and hence into nature, but is highly predictive for an input x' to provide the correct response y'. This is a common setting in machine learning and, in particular, deep learning and is illustrated by the dashed path in Figure 1.1. Because we do not have an explicit parametric model, these approaches never open the *black box* of the unknown function and thus do not enable us to examine its properties directly. A common example of these black box models are artificial neural networks which are powerful models to extract statistics and learn useful representations, as they are universal function approximators, fundamentally (Cybenko, 1989; Hornik, 1991).

In recent years, machine learning and deep learning methods were able to achieve remarkable results in diverse domains like computer vision (Krizhevsky et al., 2012; Dosovitskiy et al., 2021), natural language processing (Socher et al., 2011; Devlin et al., 2018; Brown et al., 2020), and speech recognition (Graves et al., 2013; Zhang et al., 2018b), to name a few. They also find increasing acceptance in fields of science like physics, chemistry, or biology with relevant advances in, for example, finding (surrogate) solutions to partial differential equations (Karniadakis et al., 2021), drug development (Chen et al., 2018a; Gómez-Bombarelli et al., 2018), or protein folding prediction (Tunyasuvunakool et al., 2021). Although these neural network approaches can often convince with predictive performance, their flexibility makes them notoriously difficult to be interpreted. Additionally, there are no clear guidelines for deciding when to use which architecture other than broad classes of architectures for particular problem settings. Commonly, the flexibility is restricted in an informed way by an *inductive bias* which encodes prior assumptions on how the algorithmic model is supposed to generalise to novel inputs. Convolutional neural networks are an example of employing such an inductive bias where convolutional filters reflect the assumption of local relationships in the data, like maintaining information about adjacent pixels in an image or adjacent time steps in time series data.

Although Breiman argues in favour of the algorithmic modelling culture, black box models come with the important drawback of reduced interpretability and explainability (Marcinkevičs and Vogt, 2020). In some cases, like classifying the content of an image, it might be acceptable not to know the exact mechanisms which led to a particular classification result and erroneous results are less relevant if the predictive accuracy is generally high. However, in more sensitive application domains like medicine or autonomous driving, erroneous predictions can have severe consequences. A certain degree of (post-hoc) understanding of how model predictions are influenced is necessary for deployment in practice (Durán and Jongsma, 2021). In other words, it appears desirable to find a bridge between the two cultures towards a more *traceable algorithmic modelling culture*.

With these introductory considerations we want to highlight the great potential of machine learning and deep learning methods in addressing challenging problems. This thesis focuses on representation learning and how to make use of deep learning to solve relevant real-world problems. By employing and further developing the ability of neural networks to learn powerful representations, (i) we address an important segmentation task in the geosciences, (ii) we include inductive biases to learn informed latent representations with improved generative capability for relevant applications in chemistry, and (iii) we reason about learning feature representations in common neural network architectures.

1.2 RESEARCH OBJECTIVES AND CONTRIBUTIONS

In this thesis, we develop and apply deep learning methods for a geoscience application as well as for an example in chemistry. Figure 1.2 illustrates the different areas of contribution in this thesis. A particular focus is put on encoder-decoder architectures learning compressed (latent) representations, as depicted in the first and second circle. The fundamental questions we address in this thesis cover:

- Can we make use of recent advances of convolutional neural networks and encoder-decoder architectures for relevant applications in the geosciences like soil degradation detection?
- How can we explicitly guide the learning of latent representation encodings in these architectures to increase generative abilities relevant to our down-stream application?
- In the context of recent advances in deep learning theory, what is the effect of increasing the size of deep neural networks on their performance and learning feature representations?

To provide insights and answers to these questions, we make the following contributions and structure the thesis as follows.

In **Chapter 2** we introduce a challenging geoscience task which has a particular application focus in this thesis.

To put our methodological contributions into context, we review fundamental topics of machine learning and the current state-of-the-art in **Chapter 3**.

We then present a deep learning approach based on an encoder-decoder architecture for efficient and scalable mapping of erosion phenomena in remotely-sensed high resolution imagery in **Chapter 4**. We show that our U-Net approach competes with state-of-the-art remote sensing methods for identifying erosion sites, but allows application to more extensive scales

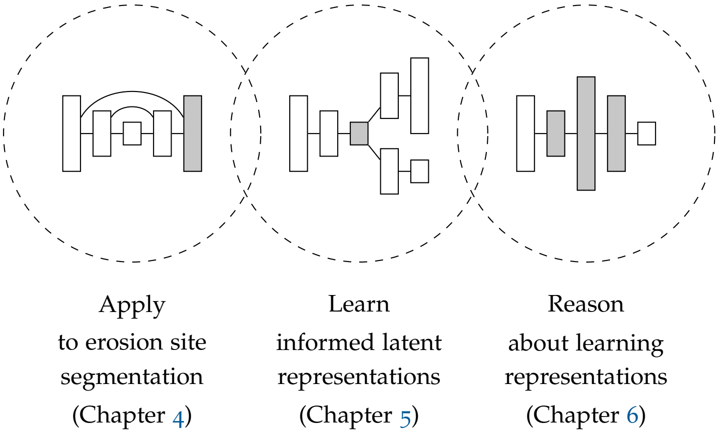


Figure 1.2: Illustration of contribution areas in this thesis. We develop and apply neural network approaches for relevant applications in geosciences and chemistry. A particular focus is on devising methods for learning informed latent representation encodings. We further reason about learning feature representations in common neural network architectures.

than feasible with established methods. The main content of this chapter is adapted from our publication Samarin et al. (2020)[2] and focuses on the Urseren valley in the Swiss Alps. Section 4.6 extends the scope and considers multiple valleys in Switzerland for learning to identify degraded soil.

Improving similar encoder-decoder architectures in the presence of relevant side information is the topic of **Chapter 5**. We introduce a novel method leveraging cycle consistency to learn informed latent encodings which enable, for instance, targeted generation of novel compounds or drugs in chemistry. The main content is adapted from our publication Samarin et al. (2021). The results in Section 5.5.3 extend the publication and study our model with respect to disentangling relevant generative factors of a dataset.

A more theoretical consideration of the learned feature representations of convolutional neural networks is conducted in **Chapter 6**. We provide an empirical study of the performance of wide neural networks and random feature models. This chapter is adapted from Samarin et al. (2022). Section

2 Part of the content was covered in the dissertation of Zweifel (2021), too.

6.5 extends these results to a different framework which allows us to consider infinitely wide neural networks.

Finally, this thesis jointly concludes our contributions in the last **Chapter 7** and provides an outlook on future directions.

1.3 LIST OF PUBLICATIONS

The following publications have been the result of some of the work presented in this thesis:

- *Mesh-free Eulerian Physics-Informed Neural Networks*.
  Fabricio Arend Torres, Marcello Negri, Monika Nagy-Huber, Maxim Samarin, and Volker Roth.
  In preparation, 2022.

- *Learning Invariances with Generalised Input-Convex Neural Networks*.
  Vitali Nesterov, Fabricio Arend Torres, Monika Nagy-Huber, Maxim Samarin, and Volker Roth.
  In preparation, 2022.

- *Feature Learning and Random Features in Standard Finite-Width Convolutional Neural Networks: An Empirical Study*.
  Maxim Samarin, Volker Roth, and David Belius.
  Conference on Uncertainty in Artificial Intelligence (UAI), 2022.

- *Learning Conditional Invariance through Cycle Consistency*.
  Maxim Samarin[3], Vitali Nesterov[3], Mario Wieser, Aleksander Wieczorek, Sonali Parbhoo, and Volker Roth.
  German Conference on Pattern Recognition (GCPR), 2021.

- *Investigating Causal Factors of Shallow Landslides in Grassland Regions of Switzerland*.
  Lauren Zweifel, Maxim Samarin, Katrin Meusburger, and Christine Alewell.
  Natural Hazards and Earth System Sciences (NHESS), 2021.

- *Learning Extremal Representations with Deep Archetypal Analysis*.
  Sebastian Keller, Maxim Samarin, Fabricio Arend Torres, Mario Wieser, and Volker Roth.
  International Journal on Computer Vision (IJCV), 2020.

3 Both authors contributed equally to this publication.

- *Identifying Soil Erosion Processes in Alpine Grasslands on Aerial Imagery with a U-Net Convolutional Neural Network*.
  Maxim Samarin[3], Lauren Zweifel[3], Volker Roth, and Christine Alewell.
  Remote Sensing, 2020.

- *Deep Archetypal Analysis*. (Honourable Mention Paper Award)
  Sebastian Keller, Maxim Samarin, Mario Wieser, and Volker Roth.
  German Conference on Pattern Recognition (GCPR), 2019.

2 SOIL DEGRADATION IN THE SWISS ALPS

A particular application focus in this thesis is put on detection of soil degradation in Swiss alpine grasslands as a challenging task with great relevance in ecology. The intactness of soil is relevant to the ecosystem as soil not only represents a habitat for microbes and wildlife, with an estimated 360 000 soil animal species (Decaëns et al., 2006), but also plays important roles in the water and nutrient cycle. Moreover, soil acts as a major carbon storage, being a larger reservoir of carbon than is contained in the atmosphere and vegetation combined (FOA, 2015). The alpine region, in particular, is a sensitive environment with its high diversity of soils, landscapes, wildlife, and vegetation (FOA, 2015). Due to their exposure to extreme climate conditions and steep terrain, Swiss alpine grassland areas can be strongly affected by soil erosion. Occurrence of erosion phenomena can have natural causes, like removal of topsoil through snow gliding, snow melt, or heavy precipitation events, to name a few, but also anthropogenic influences like disturbed vegetation due to land-use practice (Fuhrer et al., 2006; Meusburger and Alewell, 2008; Nearing et al., 2004). In case studies focusing on the Urseren valley in the Central Swiss Alps, it was shown that soil degradation has increased in the previous decades and is expected to further increase in a changing climate (Meusburger and Alewell, 2008; Zweifel et al., 2019). In that regard, findings of the *Climate Scenarios for Switzerland* (CH2018) indicate that this development can be expected to exacerbate on a larger scale. Historically, the average temperature in Switzerland has increased by 1.5°C between the pre-industrial era 1864 – 1900 and the reference period 1981 – 2010 and is expected to further increase by at least 0.6 – 1.9°C towards the end of this century (CH2018, p. 61). This projection already assumes immediate concerted actions to reduce CO_2 emission to virtually zero. At the same time, mean precipitation over Switzerland is expected to increase in winter and decrease during summer (CH2018, p. 59), while the sum of winter snowfall and surface snow cover are expected to decrease (CH2018, p. 63). These changes in precipitation and snowfall dynamics in connection with increasing temperature are believed to accelerate snow melt and reduce surface snow. Additionally, more frequent and more intense heavy precipitation events are to be expected, particularly in winter

(CH2018, p. 126). This development is expected to heavily affect alpine soils (Meusburger and Alewell, 2014).

With this perspective, a reliable quantification of soil degradation on large scales is crucial. However, considering, for instance, half of Switzerland as relevant for studying soil degradation in alpine grasslands, not only the large spatial extent but also the data size of about 1 TB for a single coverage of the area make this task a challenging big data problem[1], exceeding applicability of established methods. Therefore, we develop an efficient and scalable deep learning approach for mapping erosion phenomena in high resolution aerial imagery[2] in Chapter 4. In our work, we focus on the Urseren valley as our main case study region[3] and consider the following four erosion classes outlined in more detail in Zweifel (2021). Figure 2.1 shows examples of these erosion classes on photographs and Figure 4.3 in Chapter 4 provides examples on aerial images.

SHALLOW LANDSLIDES: This grassland erosion phenomenon is characterised by a removal of vegetation cover leading to patches of bare soil at usually steep slopes. Most shallow landslides range in size between $2 - 200$ m^2 and several examples are illustrated in Figure 2.1a. Typical causes for the occurrence are heavy precipitation events or snow cover displacement, but also slope instabilities caused by livestock.

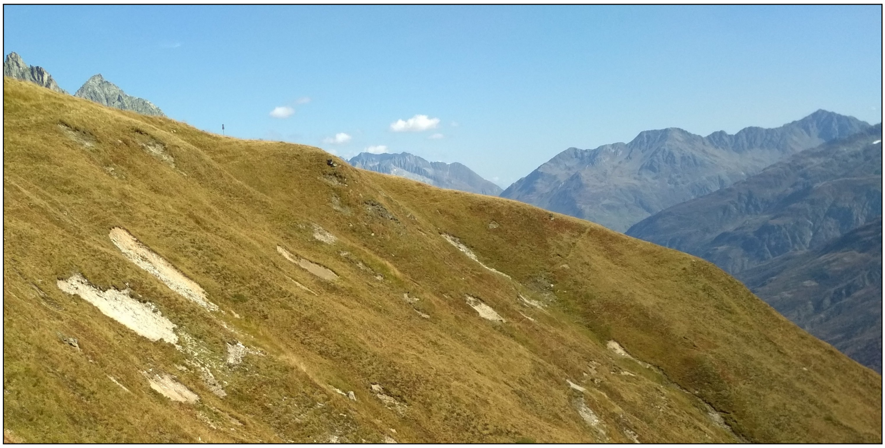
LIVESTOCK TRAILS: Livestock can further impact soil through trampling paths and leads to characteristic thin line structures typically perpendicular to slopes in steep terrain as depicted in Figure 2.1b (orange outline). Cattle traversing the pastures leads to damages of the vegetation and removal of soil which can initiate further soil degradation.

SHEET EROSION: We summarise reduced vegetation cover with often less distinct boundaries in this class. Sheet erosion describes the process of topsoil removal through water flow of usually already damaged vegetation areas. Typical causes for damages are droughts or extreme precipitation events, overgrazing and instabilities caused by livestock. In Figure 2.1b (yellow outline) more easily visible examples of sheet erosion are shown.

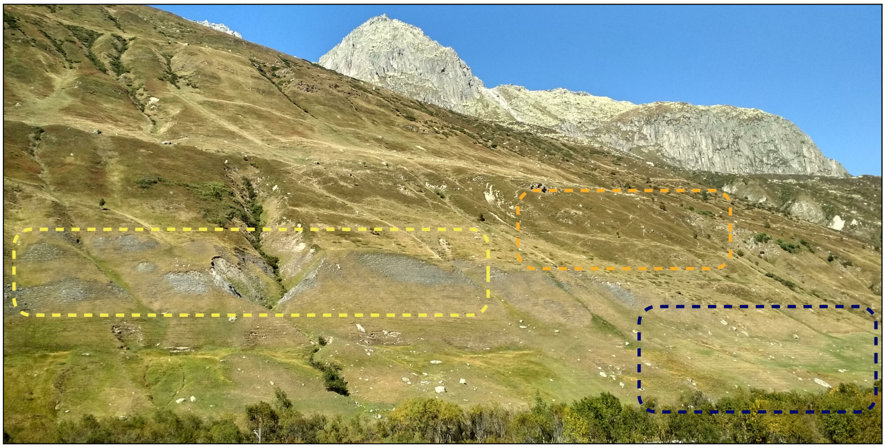
1 In this example we base the calculation on high resolution aerial imagery (RGB) of 0.25×0.25 m^2 spatial resolution.

2 This research was conducted in close collaboration with Lauren Zweifel and Christine Alewell of the Environmental Geosciences group at the University of Basel.

3 The picture on the title page shows the Urseren valley on 20th September 2018 and was recorded by the author during a field trip.



(a)



(b)

Figure 2.1: Photographs of the considered soil erosion phenomena in the Urseren valley (20th September 2018). (a) Shallow landslides. (b) Exemplary patches of livestock trails (orange), sheet erosion (yellow), and areas affected by management (blue) are highlighted. Further examples on aerial images are provided in Figure 4.3.

MANAGEMENT EFFECTS: Direct anthropogenic influence through agricultural activity affecting vegetation at typically lower elevations in the valley is summarised in the last class. Frequent examples include heavy-machinery affecting the fields, over-fertilisation, or other land-use practices which lead to reduced vegetation cover with more distinguishable geometric patterns standing out against the surrounding vegetation. Similar to the previous class, we expect that sheet erosion promotes soil erosion in this class as the dominant erosion process. Furthermore, sites affected by management display a strong seasonal dependence (e.g. due to the harvest season). Figure 2.1b (blue outline) shows such an example of grassland affected by management which led to distinct parallel lines.

3 FUNDAMENTAL CONCEPTS OF MACHINE LEARNING

In this chapter, we review the most relevant fundamental concepts of machine learning necessary for the subsequent chapters and summarise some of the state-of-the-art results in current deep learning research. Related work more closely connected to the individual contributions is covered in more detail in the respective chapters. We base the presentation on the standard machine learning textbooks by Bishop (2006) and Murphy (2012), as well as the kernel textbook by Schölkopf and Smola (2002) and we consider the textbook by Ye (2022) which covers more recent developments in deep learning. In the following, we put a larger focus on regression tasks. In the area of supervised machine learning, two types of tasks are prevalent: classification and regression. In both, we attempt to identify functional relationships between inputs and either categorical (classification) or real-valued outputs (regression). Another important area is unsupervised machine learning which is concerned with knowledge discovery like clustering, dimensionality reduction, and discovering latent factors. We touch upon this topic in context of latent variable models. In the following sections, we start with more basic machine learning topics and motivate kernel methods from linear models, review Gaussian processes, and identify their connections to deep learning. We then discuss relevant encoder-decoder architectures in context of deep latent variable models and semantic segmentation.

3.1 FROM LINEAR MODELS TO KERNEL METHODS

A common starting point to theoretically study a phenomenon is to simplify the problem setting and consider a linear model for the problem at hand. We first review the basics of linear models and then see how kernels extend these models to the non-linear setting.

3.1.1 *Linear Regression*

In many domains, a typical task consists in regression of the responses $y \in \mathbb{R}$ against the predictors or covariates $\boldsymbol{x} \in \mathbb{R}^{d_x}$. For a (training) dataset

$\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$, let us denote by $\boldsymbol{X} \in \mathbb{R}^{N \times d_x}$ the design matrix collecting N independent variables $\boldsymbol{x_n}, n \in \{1, ..., N\}$, in the rows of this matrix and by $\boldsymbol{Y} \in \mathbb{R}^N$ the corresponding responses for one-dimensional dependent variables y_n. A simple linear regression model is then given by

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{\epsilon} \tag{3.1}$$

with model parameters or weights $\boldsymbol{w} \in \mathbb{R}^{d_x}$.[1] We commonly assume that the residual error or noise term $\boldsymbol{\epsilon} \in \mathbb{R}^N$ is sampled from a Gaussian distribution, i.e. $\epsilon_n \sim \mathcal{N}(0, \sigma^2_{\text{noise}})$. An approach to solve Equation (3.1) is given by ordinary least squares (OLS) which makes use of the ℓ_2-norm or the mean squared error (MSE) loss

$$J(\boldsymbol{w}) = \frac{1}{2N}\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 = \frac{1}{2N}\sum_{n=1}^{N}(y_n - \langle \boldsymbol{x}_n, \boldsymbol{w}\rangle)^2 \tag{3.2}$$

where $\langle \boldsymbol{x}, \boldsymbol{w}\rangle = \sum_{i=1}^{d_x} x_i w_i$ denotes the scalar or inner product and $f(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{w}\rangle$ with $f\colon \mathbb{R}^{d_x} \to \mathbb{R}$ is the linear model. Minimising this objective function leads to

$$\nabla J(\boldsymbol{w}) = \boldsymbol{X}^\top \boldsymbol{Y} - \boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{w} \equiv 0, \tag{3.3}$$

$$\boldsymbol{w}_{\text{OLS}} = \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{Y}. \tag{3.4}$$

Is Equation (3.2) a good choice for an objective function if we assume a linear relationship between predictors and responses? By the Gauss-Markov theorem, we can indeed show that the parameters $\boldsymbol{w}_{\text{OLS}}$ provide the best linear unbiased estimator compared to any other linear method.

Theorem 1 (Gauss-Markov Theorem) *For the linear regression given in Equation (3.1) with design matrix $\boldsymbol{X}$, response vector $\boldsymbol{Y}$, and noise vector $\boldsymbol{\epsilon}$, the least squares estimator $\boldsymbol{w}_{OLS}$ provides the minimum-variance linear unbiased estimator of the model parameters $\boldsymbol{w}$, if and only if the following assumptions are fulfilled: (i) $\mathbb{E}[\epsilon_n] = 0$, (ii) $Var(\epsilon_n) = \sigma^2_{noise} < \infty$, (iii) $Cov(\epsilon_k, \epsilon_l) = 0, \forall n, k \neq l \in \{1, ..., N\}$, and (iv) $\boldsymbol{X}$ has full rank.*

We refer to Johnson and Wichern (1992) for a proof of this theorem. Why would we require any other linear approach? One reason is that not all assumptions might be fulfilled in practice. Typical examples are that (ii)

1 To simplify the notation, we drop the additional bias term w_0 in Equation (3.1) and in the following.

noise homoscedasticity, i.e. same finite variance, is violated or that (iv) $\boldsymbol{X}$ is not of full rank due to collinearity in the matrix, i.e. correlation between two predictor vectors $\boldsymbol{x}_k$ and $\boldsymbol{x}_l$. Another reason is that we might want to reduce variance in our solution by increasing the bias, in other words prefer a biased estimator. To see this, let us consider the expected value for the MSE of the parameter estimate $\hat{\boldsymbol{w}}$ from the optimal parameters $\boldsymbol{w}^\star$. Under the true data distribution $p(\mathcal{D}|\boldsymbol{w}^\star)$, we can calculate expected values. Let us further denote by $\bar{\boldsymbol{w}} = \mathbb{E}[\hat{\boldsymbol{w}}]$ the expected value of the parameter estimate under different datasets $\mathcal{D}$. This allows us to derive the classical bias-variance trade-off (Geman et al., 1992; Murphy, 2012):

$$\begin{aligned}
\mathbb{E}\left[(\hat{\boldsymbol{w}} - \boldsymbol{w}^\star)^2\right] &= \mathbb{E}\left[((\hat{\boldsymbol{w}} - \bar{\boldsymbol{w}}) + (\bar{\boldsymbol{w}} - \boldsymbol{w}^\star))^2\right] && (3.5)\\
&= \mathbb{E}\left[(\hat{\boldsymbol{w}} - \bar{\boldsymbol{w}})^2\right] + 2\,(\bar{\boldsymbol{w}} - \boldsymbol{w}^\star)\,\mathbb{E}\,[\hat{\boldsymbol{w}} - \bar{\boldsymbol{w}}] + (\bar{\boldsymbol{w}} - \boldsymbol{w}^\star)^2 && (3.6)\\
&= \mathbb{E}\left[(\hat{\boldsymbol{w}} - \bar{\boldsymbol{w}})^2\right] + (\bar{\boldsymbol{w}} - \boldsymbol{w}^\star)^2 && (3.7)
\end{aligned}$$

The first term in Equation (3.7) is the variance of estimator $\hat{\boldsymbol{w}}$, while the second term is the squared mean deviation from the optimal parameters or the squared bias of estimator $\hat{\boldsymbol{w}}$. We will review the bias-variance trade-off in context of highly over-parametrised neural networks in more detail (see Figure 3.3).

If our goal is to reduce the MSE and variance, we might be willing to choose a biased estimator. An approach in this direction is ridge regression[2] which penalises the ℓ_2-norm of parameters $\boldsymbol{w}$. With restricting the norm we regularise the solution $\boldsymbol{w}_{\text{ridge}}$ by diminishing over-fitting of the solution to training data and reduce the variance, i.e. sensitivity to small changes in the training data. Adding a regularisation term to the previous objective in Equation (3.2) leads to the ridge regression objective

$$\begin{aligned}
J(\boldsymbol{w}) &= \frac{1}{2N}\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \frac{1}{2}\lambda'\|\boldsymbol{w}\|_2^2 && (3.8)\\
&= \frac{1}{2N}\sum_{n=1}^{N}(y_n - \langle \boldsymbol{x}_n, \boldsymbol{w}\rangle)^2 + \frac{1}{2}\lambda'\sum_{i=1}^{d_x} w_i^2 && (3.9)
\end{aligned}$$

and the corresponding solution

$$\boldsymbol{w}_{\text{ridge}} = \left(\lambda \mathbb{1}_{d_x} + \boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{Y}. \qquad (3.10)$$

2 Also known as Tikhonov regularisation.

The effective regularisation is controlled by the Lagrange hyperparameter[3] λ and $\mathbb{1}_{d_x}$ denotes the d_x-dimensional identity matrix. This kind of regularisation is also referred to as ℓ_2-regularisation or weight decay, as the weight norm is biased towards smaller values. In case of (multi-)collinearity or linear dependencies in $\boldsymbol{X}$, we notice that $\boldsymbol{X}^\top \boldsymbol{X}$ becomes singular and thus non-invertible. Adding small values λ on the *ridge* of the first term in Equation (3.10) thus stabilises the inversion from a numerical perspective. There are also other regularisation approaches, of which the Least Absolute Shrinkage and Selection Operator (LASSO) is particularly popular one (Tibshirani, 1996). The LASSO employs the ℓ_1-norm, i.e. the absolute value, which additionally allows variable selection as parameters w_i are forced to zero.

3.1.2 *Bayesian Perspective on Ridge Regression*

We can motivate Equation (3.8) also from a Bayesian perspective and the Gaussian model for the likelihood

$$p(\boldsymbol{Y} \mid \boldsymbol{w}) = \mathcal{N}\left(\boldsymbol{X}\boldsymbol{w}, \sigma^2_{\text{noise}}\mathbb{1}_{d_x}\right) = \frac{\exp\left(-\frac{1}{2}\sigma^{-2}_{\text{noise}}\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}\|_2^2\right)}{\sqrt{(2\pi\sigma^2_{\text{noise}})^{d_x}}} \tag{3.11}$$

conditioned on the model parameters $\boldsymbol{w}$. By taking the negative log-likelihood,

$$-\log\left(\mathcal{L}\left(\boldsymbol{w} \mid \boldsymbol{Y}\right)\right) = \frac{\frac{1}{2}\sigma^{-2}_{\text{noise}}\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}\|_2^2}{\log\left(\sqrt{(2\pi\sigma^2_{\text{noise}})^{d_x}}\right)} = C \cdot \frac{1}{2}\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}\|_2^2, \tag{3.12}$$

we obtain the correspondence between the negative log-likelihood and objective function of OLS in Equation (3.2) up to the normalisation for sample size N and constant terms C, which, however, do not play any role in the minimisation. In other words, $\boldsymbol{w}_{\text{OLS}}$ is the maximum likelihood solution. If we further assume a Gaussian distribution for the prior distribution over the parameters, i.e. $p(\boldsymbol{w}) = \mathcal{N}\left(0, \sigma_w^2 \mathbb{1}_{d_x}\right)$, we can make use of Bayes' theorem to obtain the posterior distribution

$$p(\boldsymbol{w} \mid \mathbf{Y}) = \frac{p(\mathbf{Y} \mid \boldsymbol{w})p(\boldsymbol{w})}{p(\boldsymbol{Y})} \propto p(\mathbf{Y} \mid \boldsymbol{w})p(\boldsymbol{w}). \tag{3.13}$$

3 On a technical note, we require $\lambda = N\lambda'$ for consistency of notation.

To arrive at the maximum a posteriori (MAP) estimate, we use

$$-\log\left(p(\boldsymbol{w} \mid \mathbf{Y})\right) \propto \frac{1}{2}\sigma_{\text{noise}}^{-2}\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \frac{1}{2}\sigma_{\text{w}}^{-2}\|\boldsymbol{w}\|_2^2 \tag{3.14}$$

and rewrite the objective for the minimisation as

$$\frac{1}{2}\|\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{w}\|_2^2 + \frac{1}{2}\lambda\|\boldsymbol{w}\|_2^2 \tag{3.15}$$

with $\lambda = \sigma_{\text{noise}}^2/\sigma_w^2$. This corresponds to the objective function of ridge regression in Equation (3.8). As we have seen, $\boldsymbol{w}_{\text{ridge}}$ is the MAP estimate for the assumed Gaussian likelihood and prior.

3.1.3 *Kernel Methods*

However, in most cases the simplification to linear models severely limits the hypothesis class of functions relevant to our problem setting. A common approach in machine learning is to address this limitation by considering a mapping of the input space into a higher-dimensional feature space. Figure 3.1 illustrates the principal idea on a classification example. In cases where a linear model is not sufficient, we can *lift* the problem into a higher-dimensional representation. In feature space we can make use of the established linear approach, which corresponds to a non-linear model in the original input space. Coming back to the regression setting, we lift

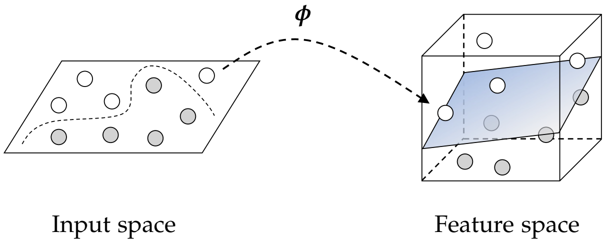


Figure 3.1: Feature mapping $\boldsymbol{\phi}$ maps data points from the input space into a higher-dimensional feature space. A linear model separates two classes in feature space for which a more complex decision boundary in input space would have been required.

predictors $\boldsymbol{x}_n$ with the feature mapping $\boldsymbol{\phi}$: $\mathbb{R}^{d_x} \to \mathbb{R}^p, \boldsymbol{x}_n \mapsto \boldsymbol{\phi}(\boldsymbol{x}_n)$ into a p-dimensional feature space. We rewrite the linear regression model of Equation (3.1) to

$$\boldsymbol{Y} = \boldsymbol{\Phi}\left(\boldsymbol{X}\right)\boldsymbol{w} + \boldsymbol{\epsilon} \tag{3.16}$$

where $\boldsymbol{w} \in \mathbb{R}^p$ and $\boldsymbol{\Phi} \in \mathbb{R}^{N\times p}$ is the feature matrix. Considering again the same objective function as in Equation (3.8) but this time for kernel ridge regression (KRR), the (MAP) solution is given by

$$\begin{aligned}
\boldsymbol{w}_{\text{KRR}} &= \left(\lambda\mathbb{1}_p + \boldsymbol{\Phi}\left(\boldsymbol{X}\right)^\top\boldsymbol{\Phi}\left(\boldsymbol{X}\right)\right)^{-1}\boldsymbol{\Phi}\left(\boldsymbol{X}\right)^\top\boldsymbol{Y} & (3.17)\\
&= \boldsymbol{\Phi}\left(\boldsymbol{X}\right)^\top\left(\boldsymbol{\Phi}\left(\boldsymbol{X}\right)\boldsymbol{\Phi}\left(\boldsymbol{X}\right)^\top + \lambda\mathbb{1}_N\right)^{-1}\boldsymbol{Y} & (3.18)\\
&= \boldsymbol{\Phi}\left(\boldsymbol{X}\right)^\top\boldsymbol{\alpha} = \sum_{n=1}^{N}\alpha_n\boldsymbol{\phi}(\boldsymbol{x}_n) & (3.19)
\end{aligned}$$

where we introduce the dual variable $\boldsymbol{\alpha} = \left(\boldsymbol{\Phi}\left(\boldsymbol{X}\right)\boldsymbol{\Phi}\left(\boldsymbol{X}\right)^\top + \lambda\mathbb{1}_N\right)^{-1}\boldsymbol{Y}$, i.e. $\boldsymbol{\alpha} \in \mathbb{R}^N$. Note that in Equation (3.18) we use a matrix identity to change the size of the matrix to be inverted (Murphy, 2012). Depending on which of the two dimensions, i.e. the number of samples N or number of features p, is larger, one or the other computation is more favourable numerically. This is important because we might even use an infinite-dimensional feature space. The insight of Equation (3.19) reveals that the solution for $\boldsymbol{w}_{\text{KRR}}$ lies in the span of the data samples even if the feature space is much larger, i.e. $N \ll p$, or possibly infinite. But is a large, potentially infinite-dimensional feature space beneficial? The brilliance of this approach comes with the observation that we never need to evaluate the feature mapping $\boldsymbol{\phi}$ directly, which is known as the kernel trick. Let us consider that we have an infinite-dimensional feature space with $p = \infty$ in the following. In order to predict the response for a new test point $\boldsymbol{x} \in \mathbb{R}^{d_x}$, we evaluate

$$\begin{aligned}
f(\boldsymbol{x}) &= \langle\boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{w}_{\text{KRR}}\rangle & (3.20)\\
&= \boldsymbol{\phi}(\boldsymbol{x})^\top\boldsymbol{\Phi}\left(\boldsymbol{X}\right)^\top\left(\boldsymbol{\Phi}\left(\boldsymbol{X}\right)\boldsymbol{\Phi}\left(\boldsymbol{X}\right)^\top + \lambda\mathbb{1}_N\right)^{-1}\boldsymbol{Y} & (3.21)\\
&= \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X})\left(\boldsymbol{K} + \lambda\mathbb{1}_N\right)^{-1}\boldsymbol{Y} & (3.22)
\end{aligned}$$

with kernel $\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m) = \langle\boldsymbol{\phi}(\boldsymbol{x}_n), \boldsymbol{\phi}(\boldsymbol{x}_m)\rangle$, kernel matrix $\boldsymbol{K} \in \mathbb{R}^{N\times N}$ with $K_{n,m} = \kappa(\boldsymbol{x}_n, \boldsymbol{x}_m)$, $\forall n, m \in \{1, .., N\}$, and $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X}) = [\kappa(\boldsymbol{x}, \boldsymbol{x}_1), ..., \kappa(\boldsymbol{x}, \boldsymbol{x}_N)]^\top$. Instead of explicitly calculating the feature vectors $\boldsymbol{\phi}(\boldsymbol{x}_n)$, it is sufficient to evaluate kernels $\kappa(\boldsymbol{x}_n, \boldsymbol{x}_m)$ representing the inner products of feature vectors.

Note that this is a completely non-parametric approach and data points are directly represented by the kernel matrix. Another way to write this is by combing Equations (3.19) and (3.20) to

$$f(\boldsymbol{x}) = \left\langle \boldsymbol{\phi}(\boldsymbol{x}), \sum_{n=1}^{N} \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n) \right\rangle = \sum_{n=1}^{N} \alpha_n \left\langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{x}_n) \right\rangle = \sum_{n=1}^{N} \alpha_n \kappa\left(\boldsymbol{x}, \boldsymbol{x}_n\right) \quad (3.23)$$

where we make use of the dual variable α_n and kernel $\kappa(\boldsymbol{x}, \boldsymbol{x}_n)$ only. This important results is known as the representer theorem. The significance of this theorem is that even for infinite-dimensional feature mappings $\boldsymbol{\phi}$, the solution lies in the span or linear combination of N kernels. In order to state the theorem more formally, we require the subsequent definitions and base the derivation on Schölkopf and Smola (2002). In the following, let $\mathcal{X}$ denote a non-empty set.

Definition 1 (Kernel) *The function $\kappa\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called kernel if there exists an inner product space $\mathcal{H}$ and a feature mapping $\boldsymbol{\phi}\colon \mathcal{X} \to \mathcal{H}$ such that $\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$*

$$\kappa\left(\boldsymbol{x}, \boldsymbol{x}'\right) = \left\langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{x}') \right\rangle_{\mathcal{H}}. \quad (3.24)$$

Definition 2 (Positive Definiteness) *We call a symmetric function $\kappa\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ positive definite if $\forall \boldsymbol{x}_n, \boldsymbol{x}_m \in \mathcal{X}$ and $\forall \alpha_n, \alpha_m \in \mathbb{R}$*

$$\sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m \kappa\left(\boldsymbol{x}_n, \boldsymbol{x}_m\right) \geqslant 0. \quad (3.25)$$

Furthermore, let us identify $\mathcal{H}$ as a Hilbert space which is a real or complex inner product space and is a complete metric space, i.e. every Cauchy sequence of points in $\mathcal{H}$ has its limit in $\mathcal{H}$. With this, we can more directly specify the Hilbert space in which kernels operate on.

Definition 3 (Reproducing Kernel Hilbert Space) *Let $\kappa\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive definite kernel and $\mathcal{H}$ a Hilbert space of functions $f\colon \mathcal{X} \to \mathbb{R}$. We call κ a reproducing kernel of $\mathcal{H}$ and $\mathcal{H}$ a reproducing kernel Hilbert space, if $\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$: (i) κ has the reproducing property, i.e. $\forall f \in \mathcal{H}$: $f(\boldsymbol{x}) = \langle f, \kappa(\cdot, \boldsymbol{x})\rangle_{\mathcal{H}}$, and in particular $\langle \kappa(\cdot, \boldsymbol{x}), \kappa(\cdot, \boldsymbol{x}')\rangle_{\mathcal{H}} = \kappa(\boldsymbol{x}, \boldsymbol{x}')$, as well as (ii) $\mathcal{H}$ is a linear span of $\{\kappa\left(\cdot, \boldsymbol{x}\right) : \boldsymbol{x} \in \mathcal{X}\}$.*

Note that $f \in \mathcal{H}$ can be viewed as the infinite vector of function coefficients while $f(\boldsymbol{x}) \in \mathcal{R}$ is the evaluation of the function at point $\boldsymbol{x} \in \mathcal{X}$. The last definition further implies that the kernel is symmetric in its arguments, that

$\kappa(\cdot, \boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x}) \in \mathcal{H}$ is a valid feature mapping and both the (centered) kernel $\kappa(\cdot, \boldsymbol{x})$ and $\boldsymbol{\phi}(\boldsymbol{x})$ are functions. We denote the norm induced by the inner product of the reproducing kernel Hilbert space (RKHS) as $\|f\|_{\mathcal{H}}$ with

$$\|f\|_{\mathcal{H}}^2 = \langle f, f \rangle_{\mathcal{H}} = \sum_{l=1}^{\infty} f_l^2 < \infty. \tag{3.26}$$

With this, we are ready to state the main theorem.

Theorem 2 (Representer Theorem) *Let $\kappa \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a positive definite kernel with corresponding RKHS $\mathcal{H}$. Let $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ be the training dataset with $\boldsymbol{x}_n \in \mathcal{X}$ and $y_n \in \mathbb{R}$. Let $\mathcal{R} \colon [0, \infty) \to \mathbb{R}$ be a strictly monotonic increasing regularisation function and $\alpha_n \in \mathbb{R}$, $n \in \{1, .., N\}$. Then for an arbitrary loss function $J \colon \left(\mathcal{X} \times \mathbb{R}^2\right)^N \to \mathbb{R} \cup \{\infty\}$, any minimiser $f \in \mathcal{H}$ for the regularised optimisation problem*

$$f^\star = \underset{f \in \mathcal{H}}{argmin}\; J\left(\{(\boldsymbol{x}_n, y_n, f(\boldsymbol{x}_n))\}_{n=1}^N\right) + \mathcal{R}\left(\|f\|_{\mathcal{H}}\right) \tag{3.27}$$

admits a representation of the form

$$f^\star(\boldsymbol{x}) = \sum_{n=1}^N \alpha_n \kappa\left(\boldsymbol{x}, \boldsymbol{x}_n\right). \tag{3.28}$$

We refer to Schölkopf and Smola (2002) for the proof. The representer theorem provides a mathematical basis for kernel methods and shows that the solution in Equation (3.23) which we obtained for the MSE loss and ridge regularisation applies more generally to any loss J and regulariser $\mathcal{R}$. Importantly, the theorem enables using kernels corresponding to an infinite number of feature mappings or basis functions. For instance, the popular radial basis function (RBF) kernel

$$\kappa\left(\boldsymbol{x}, \boldsymbol{x}'\right) = \exp\left(-\frac{1}{2\sigma_{\mathrm{bw}}^2}\|\boldsymbol{x} - \boldsymbol{x}'\|_2^2\right), \tag{3.29}$$

where σ_{bw}^2 is the bandwidth parameter, can be shown to be formed by an infinite number of feature mappings involving Hermite polynomials (Rasmussen and Williams, 2006). But we only need to evaluate the kernel in input space to make use of these infinite features. Furthermore, kernels can be interpreted as similarity measures of data points in the input space.

Thus, engineering kernels for devising task-relevant similarity measures or feature representations is a key task in working with kernel methods. Techniques for constructing valid kernels are, for example, provided in Chapter 6.2 of Bishop (2006). Although the kernel trick allows us to lift our problem into a higher-dimensional feature space without the necessity to operate in this space directly, an important limitation of kernel methods is storing the kernel matrix $\boldsymbol{K}$ which scales quadratically with the dataset size N.

3.2 GAUSSIAN PROCESS

There exists a close connection of what we have seen so far to Gaussian processes, on which we elaborate in this section. Informally, a Gaussian process extends the notion of random variables to random functions, with a more formal definition given in the following.

Definition 4 (Gaussian Process) *Let T be an arbitrary (finite) index set. We call the stochastic process $\{X(t) \mid t \in T\}$ a Gaussian process if $\forall N \in \mathbb{N}$ and $\{t_1, \ldots, t_N\} \subset T$, the random variables $(X(t_1), \ldots, X(t_N))$ have a joint Gaussian distribution.*

In our regression setting, the random variables are the function values $f(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathbb{R}^{d_x}$ with $f \colon \mathbb{R}^{d_x} \to \mathbb{R}$ as before. A Gaussian process (GP) is fully specified by its mean function $m(\boldsymbol{x})$ and its (positive definite) covariance function $\kappa(\boldsymbol{x}, \boldsymbol{x}')$ and we write the GP as

$$f(\boldsymbol{x}) \sim \mathcal{N}\left(m(\boldsymbol{x}), \kappa(\boldsymbol{x}, \boldsymbol{x}')\right) \tag{3.30}$$

with

$$\begin{aligned} m(\boldsymbol{x}) &= \mathbb{E}\left[f(\boldsymbol{x})\right], & (3.31) \\ \kappa(\boldsymbol{x}, \boldsymbol{x}') &= \mathbb{E}\left[\left(f(\boldsymbol{x}) - m(\boldsymbol{x})\right)\left(f(\boldsymbol{x}') - m(\boldsymbol{x}')\right)\right]. & (3.32) \end{aligned}$$

For the training set $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$ with design matrix $\boldsymbol{X} \in \mathbb{R}^{N \times d_x}$ and response vector $\boldsymbol{Y} \in \mathbb{R}^{N}$ as before, we denote the mean vector as $\boldsymbol{\mu} = [m(\boldsymbol{x}_1), \ldots, m(\boldsymbol{x}_N)]^\top$ and the covariance matrix as $\boldsymbol{K}$ with $K_{n,m} = \kappa(\mathbf{x}_n, \mathbf{x}_m)$, $n, m \in \{1, \ldots, N\}$. This leads to

$$(f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{K}) \tag{3.33}$$

which provides the GP prior distribution $p\left(\boldsymbol{f} \mid \boldsymbol{X}\right) = \mathcal{N}\left(\boldsymbol{\mu}, \boldsymbol{K}\right)$ for the regression function $\boldsymbol{f}\left(\boldsymbol{X}\right) = \left[f(\boldsymbol{x}_1), ..., f(\boldsymbol{x}_N)\right]^\top$. Without loss of generality, we can assume $m(\boldsymbol{x}) = 0$ as the flexibility of GPs allows modelling the mean arbitrarily well (Murphy, 2012). Similarly to Equations (3.1) and (3.16), we consider a GP regression model with noisy responses

$$\boldsymbol{Y} = \boldsymbol{f}(\boldsymbol{X}) + \boldsymbol{\epsilon} \tag{3.34}$$

and $\epsilon \sim \mathcal{N}(0, \sigma^2_{\text{noise}})$. For predicting responses $\boldsymbol{f}_*$ for a test set matrix $\boldsymbol{X}_*$, we compute the posterior predictive distribution $p(\boldsymbol{f}_* \mid \boldsymbol{X}_*, \boldsymbol{X}, \boldsymbol{Y})$. By definition, the joint prior distribution is given by

$$\begin{pmatrix} \boldsymbol{Y} \\ \boldsymbol{f}_* \end{pmatrix} \sim \mathcal{N}\left(0, \begin{pmatrix} \boldsymbol{K} + \sigma^2_{\text{noise}}\mathbb{1}_N & \boldsymbol{K}_* \\ \boldsymbol{K}_*^\top & \boldsymbol{K}_{**} \end{pmatrix}\right) \tag{3.35}$$

with $\mathbf{K}_* = \kappa(\boldsymbol{X}_*, \boldsymbol{X})$ and $\mathbf{K}_{**} = \kappa(\boldsymbol{X}_*, \boldsymbol{X}_*)$.[4] We make use of the favourable properties of Gaussian distributions (see e.g. the Gaussian identities in Rasmussen and Williams (2006)) and we see that the posterior predictive distribution $p(\boldsymbol{f}_* \mid \boldsymbol{X}_*, \boldsymbol{X}, \boldsymbol{Y})$ conditioned on the observations has the following form:

$$p(\boldsymbol{f}_* \mid \boldsymbol{X}_*, \boldsymbol{X}, \boldsymbol{Y}) = \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{K}_*) \tag{3.36}$$

$$\boldsymbol{\mu}_* = \boldsymbol{K}_* \left(\boldsymbol{K} + \sigma^2_{\text{noise}}\mathbb{1}_N\right)^{-1} \boldsymbol{Y} \tag{3.37}$$

$$\boldsymbol{K}_* = \boldsymbol{K}_{**} - \boldsymbol{K}_*^\top \left(\boldsymbol{K} + \sigma^2_{\text{noise}}\mathbb{1}_N\right)^{-1} \boldsymbol{K}_* \tag{3.38}$$

We notice that the posterior mean $\boldsymbol{\mu}_*$ in Equation (3.37) corresponds to the kernel ridge regression result in Equation (3.22) with $\lambda \equiv \sigma^2_{\text{noise}}$. More explicitly, if we consider only one test point $\boldsymbol{x}_*$, we get

$$\mu_* = \boldsymbol{K}\left(\boldsymbol{x}_*, \boldsymbol{X}\right)\left(\boldsymbol{K} + \lambda\mathbb{1}_N\right)^{-1} \boldsymbol{Y} = \boldsymbol{K}\left(\boldsymbol{x}_*, \boldsymbol{X}\right)\boldsymbol{\alpha} = \sum_{n=1}^{N} \alpha_n \kappa\left(\boldsymbol{x}_*, \boldsymbol{x}_n\right) \tag{3.39}$$

which retrieves the representer theorem in Equation (3.28), using the dual variable $\boldsymbol{\alpha} = \left(\boldsymbol{K} + \lambda\mathbb{1}_N\right)^{-1} \boldsymbol{Y}$ as in Equation (3.19). In other words, we obtain the same result with a Bayesian approach to kernel methods. The treatment in Sections 3.1.2 and 3.1.3 can be seen as the weight-space view on GP

4 With a slight abuse of notation, $\kappa(\boldsymbol{X}_*, \boldsymbol{X})$ indicates all covariance functions of training and testing data point pairs and similarly $\kappa(\boldsymbol{X}_*, \boldsymbol{X}_*)$ for all testing data point pairs.

regression with random variables $\boldsymbol{w}$, while in this section we considered the function-space view with random functions f. In particular, the covariance function is positive definite and thus can be viewed as a valid kernel function, which is the reason for the same notation. Additionally, the covariance enables quantifying the uncertainty of the prediction. However, as before, the kernel or covariance matrix $\boldsymbol{K}$ scales quadratically with training set size N and the matrix inversion scales cubically with N, which is an important limitation of the applicability of exact inference with Gaussian processes in practice. These computational shortcomings have motivated several advancements in the direction of approximate inference as, for example, low-rank approximations of the covariance matrix $\boldsymbol{K}$ (Liu et al., 2020).

3.3 DEEP LEARNING BASICS

We have seen that we can choose or design a feature mapping $\boldsymbol{\phi}$ or kernel κ relevant to our problem setting. Informally, deep learning might be characterised by attempting to learn the feature mapping $\boldsymbol{\phi}$ from data and thus provide adaptive basis functions. In the following, we show that kernel methods, as covered so far, and neural networks can be viewed as duals of each other. In this section, we first focus on some basics of neural networks and motivate neural network approaches more formally. In the subsequent sections, we consider infinitely-wide neural networks and their connection to Gaussian processes. Lastly, we focus on deep latent variable models and encoder-decoder architectures, as well as deep learning approaches for semantic segmentation.

3.3.1 *Multilayer Perceptron*

We motivate the general idea of neural networks in Figure 3.2. A Multilayer Perceptron (MLP), also referred to as feedforward neural network, generally consist of a combination of an input layer, one to several hidden layers, and an output layer. The illustrated three-layer MLP[5]

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{W}) = \boldsymbol{W}^{(L)\top}\boldsymbol{\sigma}(\boldsymbol{W}^{(2)\top}\boldsymbol{\sigma}(\boldsymbol{W}^{(1)\top}\boldsymbol{x})) \tag{3.40}$$

processes an input $\boldsymbol{x} \in \mathbb{R}^{d_x}$ and parametrises a function $\boldsymbol{f}\colon \mathbb{R}^{d_x} \to \mathbb{R}^{d_y}$ in a sequential manner, where $\boldsymbol{\sigma}$ is an element-wise non-linear activation func-

5 As before, we drop all additional bias terms to simplify the notation.

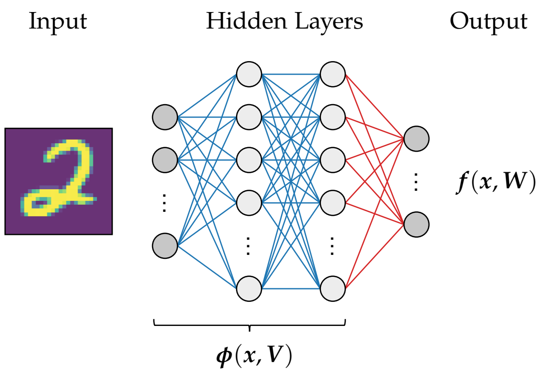


Figure 3.2: Illustration of a three-layer Multilayer Perceptron (MLP). Two hidden layers process the input and retrieve a feature mapping $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{V})$. This learned representation is used for predicting the output $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{W})$ in the last layer. Units are depicted by grey circles, edges represent parameters $w \in \boldsymbol{W}$.

tion, like the rectified linear unit (ReLU) $\sigma(\cdot) = \max\{0, \cdot\}$. The parameter or weight matrices $\{\boldsymbol{W}^{(1)}, ..., \boldsymbol{W}^{(L)}\}$, with $\boldsymbol{W}^{(l)} \in \mathbb{R}^{d_{l-1} \times d_l}$, map activations from layer $l-1$ to the subsequent layer l, with $l \in \{1, ..., L\}$ enumerating the different layers starting from the first hidden layer with $l = 1$ to the output layer with $l = L$. In a fully-connected MLP all neurons or units of one layer are connected to all units of the subsequent layer and $d_0 \equiv d_x$ as well as $d_L \equiv d_y$. Hence, d_l denotes the number of units or the width of layer l. Let us denote by $\boldsymbol{V} = \{\boldsymbol{W}^{(1)}, ..., \boldsymbol{W}^{(L-1)}\}$ the set of all parameters except for those mapping to the output layer and by $\boldsymbol{W} = \{\boldsymbol{V}, \boldsymbol{W}^{(L)}\}$ the set of all parameters. Typically, the parameters are initialised as draws from a Gaussian distribution and updated during training. To this end, the gradients with respect to a cost or loss function $J(\boldsymbol{W})$, like the MSE loss we used so far, are obtained with the backpropagation algorithm implementing the chain rule of derivatives (Bryson and Ho, 1969; Rumelhart et al., 1986). With the goal of minimising loss $J(\boldsymbol{W})$, the parameters $w \in \boldsymbol{W}$ are iteratively adjusted by optimisation methods like (stochastic) gradient descent or Adam (Kingma and Ba, 2015a).

3.3.2 *Neural Network Regression Model*

Returning to the same regression setting as before with $d_y = 1$ and noise $\epsilon \sim \mathcal{N}(0, \sigma^2_{\text{noise}})$, we can write the neural network regression model as

$$\boldsymbol{Y} = \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{W}) + \boldsymbol{\epsilon} = \boldsymbol{\Phi}(\boldsymbol{X}, \boldsymbol{V})\boldsymbol{w}^{(L)} + \boldsymbol{\epsilon} \tag{3.41}$$

where $\boldsymbol{w}^{(L)} \in \mathbb{R}^{d_{L-1}}$ and feature matrix $\boldsymbol{\Phi} \in \mathbb{R}^{N \times d_{L-1}}$, i.e. the rows correspond to the feature mapping $\boldsymbol{\phi}\colon \mathbb{R}^{d_x} \to \mathbb{R}^{d_{L-1}}$ for an individual data point $\boldsymbol{x}$. Formulating the model in this way, we obtain a correspondence to Equations (3.1), (3.16), and (3.34), but we make the dependence on the parameters $\boldsymbol{W}$ explicit as these are successively updated during the training. Therefore, we can view a neural network as learning a non-linear feature mapping $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{V})$ and employing a linear model parametrised by $\boldsymbol{w}^{(L)}$ to predict the response. More explicitly, we can formulate the MSE loss function for this linear model as

$$J(\boldsymbol{w}^{(L)}) = \frac{1}{2N} \sum_{n=1}^{N} \left(y_n - \left\langle \boldsymbol{\phi}(\boldsymbol{x}_n, \boldsymbol{V}), \boldsymbol{w}^{(L)} \right\rangle \right)^2 + \frac{1}{2}\lambda' \left\langle \boldsymbol{w}^{(L)}, \boldsymbol{w}^{(L)} \right\rangle \tag{3.42}$$

where we used weight decay or ℓ_2-regularisation of the last layer weights.[6] Minimising this objective with respect to $\boldsymbol{w}^{(L)}$ leads to the (MAP) solution[7]

$$\begin{aligned} \boldsymbol{w}_{NN} &= \frac{1}{\lambda} \sum_{n=1}^{N} \left(y_n - \left\langle \boldsymbol{\phi}(\boldsymbol{x}_n, \boldsymbol{V}), \boldsymbol{w}^{(L)} \right\rangle \right) \boldsymbol{\phi}(\boldsymbol{x}_n, \boldsymbol{V}) &(3.43)\\ &= \sum_{n=1}^{N} \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_n, \boldsymbol{V}) = \boldsymbol{\Phi}(\boldsymbol{X}, \boldsymbol{V})^\top \boldsymbol{\alpha} &(3.44) \end{aligned}$$

with dual variable $\boldsymbol{\alpha}$ and $\alpha_n = \frac{1}{\lambda}\left(y_n - \left\langle \boldsymbol{\phi}(\boldsymbol{x}_n, \boldsymbol{V}), \boldsymbol{w}^{(L)} \right\rangle \right)$. Rewriting the loss function in terms of this dual variable $\boldsymbol{\alpha}$ and simplifying the notation to $\boldsymbol{\Phi} \equiv \boldsymbol{\Phi}(\boldsymbol{X}, \boldsymbol{V})$, leads to the dual loss function

$$\begin{aligned} J(\boldsymbol{\alpha}) &= \frac{1}{2N} \sum_{n=1}^{N} \left(y_n - \left\langle \boldsymbol{\phi}(\boldsymbol{x}_n, \boldsymbol{V}), \boldsymbol{\Phi}^\top \boldsymbol{\alpha} \right\rangle \right)^2 + \frac{1}{2}\lambda' \left\langle \boldsymbol{\Phi}^\top \boldsymbol{\alpha}, \boldsymbol{\Phi}^\top \boldsymbol{\alpha} \right\rangle &(3.45)\\ &= \frac{1}{2N} \left(\boldsymbol{Y}^\top \boldsymbol{Y} - 2\boldsymbol{\alpha}^\top \boldsymbol{K}\boldsymbol{Y} + \boldsymbol{\alpha}^\top \boldsymbol{K}\boldsymbol{K}\boldsymbol{\alpha} \right) + \frac{1}{2}\lambda' \boldsymbol{\alpha}^\top \boldsymbol{K}\boldsymbol{\alpha} &(3.46) \end{aligned}$$

6 Note that in a more general setting all parameters in $\boldsymbol{W}$ are regularised, typically.
7 As before, we require $\lambda = N\lambda'$ for consistency of notation.

where we made use of the kernel matrix $\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top$ as defined in Section 3.1.3. For $\nabla J(\boldsymbol{\alpha}) = 0$, we obtain the solution to the minimisation

$$\boldsymbol{\alpha} = \left(\boldsymbol{K} + \lambda \mathbb{1}_N\right)^{-1} \boldsymbol{Y} \tag{3.47}$$

which retrieves the results for $\boldsymbol{\alpha}$ we encountered for kernel methods in Equation (3.19) and Gaussian processes in Equation (3.39). This shows the duality of kernel methods and neural networks: While neural networks parametrise and learn the (non-linear) feature mapping $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{V})$ by optimising parameters $\boldsymbol{W}$, kernel approaches only require inner products of features and solve the task by considering appropriate kernel functions $k(\boldsymbol{x}, \boldsymbol{x}')$ and the dual parameters $\boldsymbol{\alpha}$.

3.3.3 *Over-Parametrised Neural Networks*

Neural networks offer a great flexibility for learning features or representations in data-driven manner. A key component of the success of neural network approaches lies in employing models with an extensive number of parameters. Let us denote by p the number of parameters in $\boldsymbol{W}$. At odds with standard statistical learning theory (Vapnik, 1999), a common premise in deep learning is that over-parametrisation does not harm generalisation on a test set (Neyshabur et al., 2019), which is a common setting for state-of-the-art neural network architectures with millions or billions of parameters (Krizhevsky, 2014; Devlin et al., 2018; Dosovitskiy et al., 2021; Brown et al., 2020). This is due the observation that highly over-parametrised neural networks seem not to suffer from overfitting to the training set in the regime where the number of parameters p exceeds the number of samples N. As illustrated in Figure 3.3, a characteristic *double descent* behaviour in over-parametrised neural networks is observed (Spigler et al., 2019; Belkin et al., 2019; Advani et al., 2020; Nakkiran et al., 2021). Similar results can be obtained for random forests and AdaBoost (Wyner et al., 2017) in the over-parametrised regime, too.

In the classical regime of $p \lesssim N$, we observe the established bias-variance trade-off, which we derived in Equation (3.7). In this regime, we reach an optimum of intermediate model complexity p_{BV} and locally minimal generalisation error. Increasing the model complexity from this point on improves the training error but leads to an increase in the test error, thus an increasing generalisation gap or, in other words, overfitting. For $p \approx N$, we reach the interpolation threshold p_{inter} where the model is complex enough to perfectly fit every training data point. However, we observe empirical that with further increasing model complexity, i.e. over-parametrisation, the test error monotonically decreases and overfitting is reduced up to the point that highly over-parametrised models, i.e. $p \gg N$, provide lower generalisation errors overall. Note that we use the number of parameters as a typical proxy for the model complexity. More appropriate measures include training procedure aspects into the notion of model complexity like the effective model complexity (EMC) (Nakkiran et al., 2021) and are related to established notions like Rademacher complexity or Vapnik-Chervonenkis (VC) dimension. However, deriving (tight) generalisation bounds on the test error is a challenging task in deep learning (Jiang et al., 2020).

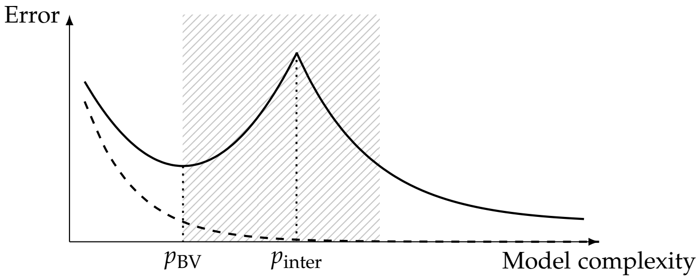


Figure 3.3: Illustration of the double descent phenomenon. The train error (dashed) and test error (solid) are plotted against the model complexity. The interpolation threshold p_{inter} (dotted) separates two regimes. The classical regime of the bias-variance trade-off is encountered below p_{inter} in which increasing model complexity can result in an increase of the generalisation error. This leads to a local optimum of intermediate model complexity p_{BV} (dotted) and locally minimal generalisation error. Above p_{inter} we obtain the modern interpolating regime where models with high model complexity lead to a monotonically decreasing generalisation error. The shaded area depicts the overfitting regime in which models with a complexity close to p_{inter} lead to overfitting.

Therefore, a fundamental difference between the classical bias-variance trade-off and the recent "*weak but plentiful features interpolating setting*" (Belkin et al., 2020) is suggested. The remarkable generalisation is achieved even without explicit regularisation (Valle-Perez et al., 2019). While most of the investigation focus on the monotonically decreasing test error, studies like Neal et al. (2018) or Hastie et al. (2022) perform a theoretical analysis for a modern take on the bias-variance trade-off. But the phenomenon is still yet to be understood. Popular lines of argument attribute these results to particular features of optimisers like SGD, the local curvature of stationary points of the loss function or that deep neural networks are intrinsically biased towards "*simple*" functions (Valle-Perez et al., 2019). In this thesis, we empirically study the generalisation performance of such highly over-parametrised neural networks in Chapter 6.

3.4 NEURAL NETWORKS IN THE INFINITE-WIDTH LIMIT

We review highly over-parametrised neural networks more closely by considering the infinite width limit. Early work by Neal (1996) and Williams (1996) in that direction showed that two-layer MLPs with randomly initialised parameters and a single hidden layer consisting of infinitely many units converge to Gaussian processes. In recent years, these results were extended to deep MLPs (de G. Matthews et al., 2018; Lee et al., 2018) as well as other deep neural network architectures like convolutional neural networks (Novak et al., 2018; Garriga-Alonso et al., 2019), recurrent neural networks (Yang, 2019a), or transformers (Hron et al., 2020). In the following, we highlight some of the main results.

3.4.1 *Neural Network Gaussian Process*

We focus on MLPs with L layers at initialisation and successively take the width of each hidden layer to infinity. The derivation is based on Lee et al. (2018). Let us denote by $\boldsymbol{z}^{(l)} = \boldsymbol{W}^{(l)\top}\boldsymbol{x}^{(l-1)}$ the pre-activations and by $\boldsymbol{x}^{(l)} = \sigma(\boldsymbol{z}^{(l)})$ the activations in layer l. We initialise the parameters as $W_{j,i}^{(l)} \sim \mathcal{N}(0, \sigma_w^2/d_{l-1})$ for layer l with d_l units and $j \in \{1, ..., d_{l-1}\}$, $i \in \{1, ..., d_l\}$. For an input $\boldsymbol{x}$ and a unit $z_i^{(l)}$ in layer l, we can write[8]

$$z_i^{(l)}(\boldsymbol{x}) = \sum_{j=1}^{d_{l-1}} W_{i,j}^{(l)} x_j^{(l-1)} \tag{3.48}$$

$$x_j^{(l-1)}(\boldsymbol{x}) = \sigma\left(z_j^{(l-1)}(\boldsymbol{x})\right) = \sigma\Big(\sum_{k=1}^{d_{l-2}} W_{j,k}^{(l-1)} x_k^{(l-2)}\Big). \tag{3.49}$$

Let us consider first the case of $l = 2$, such that $z_i^{(2)}(\boldsymbol{x})$ corresponds to the outputs $f^i(\boldsymbol{x}, \boldsymbol{W})$ of a single hidden layer MLP with $\boldsymbol{x}^{(0)} \equiv \boldsymbol{x}$ and

8 Note that we again drop the bias terms to simplify the notation. The presented derivation still holds if bias terms are included, but becomes more involved; see e.g. de G. Matthews et al. (2018) or Lee et al. (2018) for more details.

$d_0 \equiv d_x$. Because we initialise the network parameters as independent and identically distributed (i.i.d.) random variables, activations $x_j^{(1)}$ and $x_j^{(1)}$ are independent for $i \neq j$. Then $z_i^{(2)}(\boldsymbol{x})$ is distributed according to a Gaussian distribution for $d_1 \to \infty$ by the Central Limit Theorem, as it is an infinite sum of i.i.d. terms. And accordingly, the activations for all inputs $\{z_i^{(2)}(\boldsymbol{x}_1), ..., z_i^{(2)}(\boldsymbol{x}_N)\}$ have a joint Gaussian distribution, which is the definition of a Gaussian process (see Definition 4). Therefore, we obtain the GP

$$z_i^{(2)}(\boldsymbol{x}) \sim \mathcal{N}\left(m^{(2)}(\boldsymbol{x}), \kappa^{(2)}(\boldsymbol{x}, \boldsymbol{x}')\right) \tag{3.50}$$

$$m^{(2)}(\boldsymbol{x}) = \mathbb{E}\left[z_i^{(2)}(\boldsymbol{x})\right] = 0 \tag{3.51}$$

$$\kappa^{(2)}(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}\left[z_i^{(2)}(\boldsymbol{x}) z_i^{(2)}(\boldsymbol{x}')\right] = \sigma_w^2 \mathbb{E}\left[x_i^{(1)}(\boldsymbol{x}) x_i^{(1)}(\boldsymbol{x}')\right] \tag{3.52}$$

where mean $m^{(2)}$ vanishes because the parameters have zero mean. By induction, we extend the result to subsequent layers in the same manner. Consider a GP $z_i^{(l-1)}$ which is identical and independent for all $i \in \{1, ..., d_{l-1}\}$. As before, we have a sum of i.i.d. random variables such that for $d_{l-1} \to \infty$ activations $\{z_i^{(l)}(\boldsymbol{x}_1), ..., z_i^{(l)}(\boldsymbol{x}_N)\}$ are jointly Gaussian and $z_i^{(l)} \sim \mathcal{N}(0, \kappa^{(l)})$ is a GP with zero mean and covariance

$$\kappa^{(l)}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_w^2 \mathbb{E}_{z_i^{(l-1)} \sim \mathcal{N}(0, \kappa^{(l-1)})}\left[\sigma\left(z_i^{(l-1)}(\boldsymbol{x})\right) \sigma\left(z_i^{(l-1)}(\boldsymbol{x}')\right)\right]. \tag{3.53}$$

The required expected value is equivalently obtained by integrating against the joint distribution of $z_i^{(l-1)}(\boldsymbol{x})$ and $z_i^{(l-1)}(\boldsymbol{x}')$ (Lee et al., 2018), which is a two-dimensional Gaussian distribution with zero mean and covariance matrix

$$\boldsymbol{\Sigma}^{(l-1)}(\boldsymbol{x}, \boldsymbol{x}') = \begin{pmatrix} \kappa^{(l-1)}(\boldsymbol{x}, \boldsymbol{x}) & \kappa^{(l-1)}(\boldsymbol{x}, \boldsymbol{x}') \\ \kappa^{(l-1)}(\boldsymbol{x}', \boldsymbol{x}) & \kappa^{(l-1)}(\boldsymbol{x}', \boldsymbol{x}') \end{pmatrix}. \tag{3.54}$$

Thus, we may rewrite Equation (3.53) with $u = z_i^{(l-1)}(\boldsymbol{x})$ and $v = z_i^{(l-1)}(\boldsymbol{x}')$ as

$$\kappa^{(l)}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_w^2 \mathbb{E}_{(u,v) \sim \mathcal{N}(0, \boldsymbol{\Sigma}^{(l-1)})}\left[\sigma(u) \sigma(v)\right]. \tag{3.55}$$

This suggests a recursive relationship between covariance matrices $\boldsymbol{K}^{(l)}$ and $\boldsymbol{K}^{(l-1)}$ which allows calculating the final covariance $\boldsymbol{K}^{(L)}$ in an iterative fashion. For the starting point we obtain a covariance matrix with entries

$$\kappa^{(1)}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_w^2 \mathbb{E}\left[x_i^{(0)}(\boldsymbol{x}) x_i^{(0)}(\boldsymbol{x}')\right] = \sigma_w^2 \mathbb{E}\left[\langle \boldsymbol{x}, \boldsymbol{x}' \rangle\right] = \sigma_w^2 \frac{\langle \boldsymbol{x}, \boldsymbol{x}' \rangle}{d_x} \tag{3.56}$$

for centered inputs $\boldsymbol{x}, \boldsymbol{x}'$. In summary, we derived a GP prior which corresponds to an MLP with infinitely wide hidden layers. This was achieved by sequentially increasing layers to infinite width and obtaining a layer-wise GP which is for this reason referred to as Neural Network Gaussian Process (NNGP). In a slightly more informal way than presented in Equation (3.55), we can write the covariance or kernel function $\boldsymbol{K}_{\text{NNGP}}$ of the NNGP as

$$\kappa_{i,j}(\boldsymbol{x}, \boldsymbol{x}') = \lim_{\min\{d_1,\ldots,d_{L-1}\}\to\infty} \mathbb{E}\left[\left\langle f^i(\boldsymbol{x}, \boldsymbol{W}), f^j(\boldsymbol{x}', \boldsymbol{W}) \right\rangle\right] \tag{3.57}$$

where f^i and f^j are different output dimensions of output vector $\boldsymbol{f} \in \mathbb{R}^{d_y}$ which is an NNGP with $\boldsymbol{f}(\boldsymbol{X}, \boldsymbol{W}) \sim \mathcal{N}(0, \boldsymbol{K}_{\text{NNGP}})$ and the expected value is with respect to the distribution of parameters $\boldsymbol{W}$. In order to predict an output for test point $\boldsymbol{x}_*$, we can make use of the posterior predictive distribution derived in Equation (3.36). Note that the considered setting corresponds to a mostly untrained neural network as we consider the MLP at initialisation and without training. Connecting to the derivation in Section 3.3.2, we can view an NNGP as using an infinite-dimensional random feature mapping $\boldsymbol{\Phi}(\boldsymbol{X}, \boldsymbol{V})$ and performing Bayesian inference for prediction. In this vein, the NNGP can be viewed as a weakly-trained neural network in the infinite-width limit where only the last-layer-weights are adjusted.

3.4.2 *Neural Tangent Kernel*

However, the appeal of deep learning is that – by training a neural network approach – we are able to learn feature mappings relevant to our problem setting. Therefore, it is desirable to extend the insights gained in the last section to GPs corresponding to fully-trained networks. The Neural Tangent Kernel (NTK) (Jacot et al., 2018) provides an attempt in this direction and we present some of the main results of NTK theory following Lee et al. (2019).

3.4.2.1 *Training Dynamics of an MLP*

We again focus on MLPs with L layers. Different from before, we initialise the parameters as $w_{j,i}^{(l)} \sim \mathcal{N}(0,1)$ such that $W_{j,i}^{(l)} = (\sigma_w/\sqrt{d_{l-1}})w_{j,i}^{(l)}$, which is known as the *NTK parametrisation*. Let $\boldsymbol{\omega}^{(l)} \in \mathbb{R}^{d_{l-1}d_l}$ denote the vector of parameters of layer l in matrix $\boldsymbol{W}^{(l)} \in \mathbb{R}^{d_{l-1}\times d_l}$ and $\boldsymbol{\omega} \in \mathbb{R}^p$ the vector containing all parameters in $\boldsymbol{W}$. Similarly, we denote by $\boldsymbol{f}(\boldsymbol{X},\boldsymbol{\omega}) \in \mathbb{R}^{Nd_y}$ the vector of network outputs $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\omega}) \in \mathbb{R}^{d_y}$. Further, let $\boldsymbol{\omega}_t$ be the training time or iteration dependent parameters at time t and $\boldsymbol{\omega}_0$ denoting the parameters at initialisation (before training). Considering the training of the MLP under gradient flow, i.e. infinitesimal step size or learning rate η, and full batch gradient descent, the evolution of parameters $\boldsymbol{\omega}$ and outputs $\boldsymbol{f}$ is given by the ordinary differential equations (ODEs)

$$\dot{\boldsymbol{\omega}}_t = -\eta\nabla_{\boldsymbol{\omega}}\boldsymbol{f}(\boldsymbol{X},\boldsymbol{\omega}_t)^\top\nabla_{\boldsymbol{f}}J(\boldsymbol{\omega}_t), \tag{3.58}$$

$$\dot{\boldsymbol{f}}(\boldsymbol{X},\boldsymbol{\omega}_t) = \nabla_{\boldsymbol{\omega}}\boldsymbol{f}(\boldsymbol{X},\boldsymbol{\omega}_t)\dot{\boldsymbol{\omega}}_t \tag{3.59}$$

$$= -\eta\nabla_{\boldsymbol{\omega}}\boldsymbol{f}(\boldsymbol{X},\boldsymbol{\omega}_t)\nabla_{\boldsymbol{\omega}}\boldsymbol{f}(\boldsymbol{X},\boldsymbol{\omega}_t)^\top\nabla_{\boldsymbol{f}}J(\boldsymbol{\omega}_t), \tag{3.60}$$

where $\dot{\boldsymbol{\omega}}_t$ is the derivative of $\boldsymbol{\omega}_t$ with respect to time and J the loss function which is supposed to be minimised. Note that $\nabla_{\boldsymbol{\omega}}\boldsymbol{f}(\boldsymbol{X},\boldsymbol{\omega}_t) \in \mathbb{R}^{Nd_y\times p}$ denotes the Jacobian and $\nabla_{\boldsymbol{f}}J(\boldsymbol{\omega}_t) \in \mathbb{R}^{Nd_y}$ is the gradient of the loss with respect to output $\boldsymbol{f}(\boldsymbol{X},\boldsymbol{\omega}_t)$. Although we use gradient flow, we keep the dependency on hyperparameter η for later consideration. Let us denote by $\boldsymbol{\Theta}_t^{(L)} \equiv \boldsymbol{\Theta}_t^{(L)}(\boldsymbol{X},\boldsymbol{X}) \in \mathbb{R}^{Nd_y\times Nd_y}$ the *empirical finite-width neural tangent kernel* at time t with

$$\boldsymbol{\Theta}_t^{(L)}(\boldsymbol{X},\boldsymbol{X}) = \nabla_{\boldsymbol{\omega}}\boldsymbol{f}(\boldsymbol{X},\boldsymbol{\omega}_t)\nabla_{\boldsymbol{\omega}}\boldsymbol{f}(\boldsymbol{X},\boldsymbol{\omega}_t)^\top \tag{3.61}$$

$$= \sum_{l=1}^{L}\nabla_{\boldsymbol{\omega}^{(l)}}\boldsymbol{f}(\boldsymbol{X},\boldsymbol{\omega}_t)\nabla_{\boldsymbol{\omega}^{(l)}}\boldsymbol{f}(\boldsymbol{X},\boldsymbol{\omega}_t)^\top. \tag{3.62}$$

Hence, the corresponding feature mapping of data point $\boldsymbol{x}$ is given by the gradient of the network output, i.e. $\boldsymbol{\Phi}(\boldsymbol{x},\boldsymbol{\omega}_t) = \nabla_{\boldsymbol{\omega}}\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\omega}_t)$.

3.4.2.2 *Training Dynamics of a Linearised MLP*

Let us consider the linearisation of the previous MLP in weight-space and use the linear model given by the first order Taylor expansion

$$\boldsymbol{f}_{\text{lin}}(\boldsymbol{x},\boldsymbol{u}_t) = \boldsymbol{f}(\boldsymbol{x},\boldsymbol{\omega}_0) + \nabla_{\boldsymbol{\omega}}\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\omega}_0)\boldsymbol{u}_t \tag{3.63}$$

where we introduce parameters $\boldsymbol{u}_t = \boldsymbol{\omega}_t - \boldsymbol{\omega}_0$ providing the deviation of the parameters $\boldsymbol{\omega}_t$ from their initial values. In this expansion, the first term captures the output of the MLP at initialisation, while the second term provides the changes of the output during training assuming a linear model. The time evolution of this linear model is thus described by ODEs

$$\dot{\boldsymbol{u}}_t = -\eta \nabla_{\boldsymbol{\omega}} \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{\omega}_0)^\top \nabla_{f_{\text{lin}}} J(\boldsymbol{u}_t), \tag{3.64}$$

$$\dot{\boldsymbol{f}}_{\text{lin}}(\boldsymbol{X}, \boldsymbol{u}_t) = -\eta \Theta_0^{(L)} \nabla_{f_{\text{lin}}} J(\boldsymbol{u}_t). \tag{3.65}$$

For the MSE loss, we can obtain closed form solutions given by

$$\boldsymbol{u}_t = -\nabla_{\boldsymbol{\omega}} \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{\omega}_0)^\top \boldsymbol{\Theta}_0^{(L)-1} \left(\mathbb{1}_{Nd_y} - \exp(-\eta \boldsymbol{\Theta}_0^{(L)} t)\right) \left(\boldsymbol{f}(\boldsymbol{X}, \boldsymbol{\omega}_0) - \boldsymbol{Y}\right), \tag{3.66}$$

$$\boldsymbol{f}_{\text{lin}}(\boldsymbol{X}, \boldsymbol{u}_t) = \left(\mathbb{1}_{Nd_y} - \exp(-\eta \boldsymbol{\Theta}_0^{(L)} t)\right) \boldsymbol{Y} + \exp(-\eta \boldsymbol{\Theta}_0^{(L)} t) \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{\omega}_0). \tag{3.67}$$

This result highlights that the dynamics of the parameters $\boldsymbol{u}_t$ and network outputs $\boldsymbol{f}_{\text{lin}}(\boldsymbol{X}, \boldsymbol{u}_t)$ are fully described by kernel $\boldsymbol{\Theta}_0^{(L)}$ and we have explicit solutions without ever training the linearised network. In other words, we only require the random feature mapping $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{\omega}_0) = \nabla_{\boldsymbol{\omega}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\omega}_0)$ fixed at initialisation.

3.4.2.3 *Infinite-Width Limit*

The relevance of the network linearisation becomes more apparent when we consider the infinite-width limit of the standard MLP $\boldsymbol{f}$ and its linearisation $\boldsymbol{f}_{\text{lin}}$. To this end, we need the limiting *infinite-width NTK* $\boldsymbol{\Theta}$.

Theorem 3 (Neural Tangent Kernel (Jacot et al., 2018)) *For an MLP of depth L at initialisation, with a Lipschitz non-linearity σ, and in the limit as the layer widths $d_1, \ldots, d_{L-1} \to \infty$, the NTK $\Theta_t^{(L)}$ converges in probability to a deterministic limiting kernel*

$$\Theta_t^{(L)} \to \Theta_\infty^{(L)} \otimes \mathbb{1}_{d_L}. \tag{3.68}$$

The scalar kernel $\Theta_\infty^{(L)} \colon \mathbb{R}^{d_x} \times \mathbb{R}^{d_x} \to \mathbb{R}$ is defined recursively by

$$\Theta_\infty^{(1)}(\boldsymbol{x}, \boldsymbol{x}') = \kappa^{(1)}(\boldsymbol{x}, \boldsymbol{x}'), \tag{3.69}$$

$$\Theta_\infty^{(L+1)}(\boldsymbol{x}, \boldsymbol{x}') = \Theta_\infty^{(L)}(\boldsymbol{x}, \boldsymbol{x}') \dot{\kappa}^{(L+1)}(\boldsymbol{x}, \boldsymbol{x}') + \kappa^{(L+1)}(\boldsymbol{x}, \boldsymbol{x}'), \tag{3.70}$$

where

$$\kappa^{(L+1)}(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{f \sim \mathcal{N}(0, \kappa^{(L)})} \left[\sigma\left(f(\boldsymbol{x})\right) \sigma\left(f(\boldsymbol{x}')\right) \right], \tag{3.71}$$

$$\dot{\kappa}^{(L+1)}(\boldsymbol{x}, \boldsymbol{x}') = \mathbb{E}_{f \sim \mathcal{N}(0, \kappa^{(L)})} \left[\dot{\sigma}\left(f(\boldsymbol{x})\right) \dot{\sigma}\left(f(\boldsymbol{x}')\right) \right], \tag{3.72}$$

taking the expectation with respect to a centered Gaussian process f of covariance $\kappa^{(L)}$, and where $\dot{\sigma}$ denotes the derivative of σ.

We refer to Jacot et al. (2018) for the proof and more details. We want to highlight that the scaling in the NTK parametrisation is of particular relevance which ensures applicability of the law of large numbers in the proof. Note that the recursive definition of the kernel is similar to what we have seen for the NNGP (see Section 3.4.1). In particular, Equations (3.53) and (3.71) correspond to each other with $l = L + 1$ and noting that the variance in NTK parametrisation is 1 as well as that $f(\boldsymbol{x}) \equiv z^{(L)}(\boldsymbol{x})$. Furthermore, we identify $\boldsymbol{\Theta}(\boldsymbol{X}, \boldsymbol{X}) \equiv \boldsymbol{\Theta}_\infty^{(L)}(\boldsymbol{X}, \boldsymbol{X})$ and note that the activation function σ to be Lipschitz is generally fulfilled for popular choices like ReLUs. By Theorem 3, we can compute the infinite-width NTK $\boldsymbol{\Theta}$ in an iterative fashion, similar to $\boldsymbol{K}_{\text{NNGP}}$, and the exact form only depends on the network architecture and depth, the choice of activation function σ and the initialisation of parameters $\boldsymbol{\omega}$. With this in place, we can study the infinite-width behaviour of f_{lin} more closely. Note that we use a similar notation as introduced for the joint GP prior in Equation (3.35) and denote all layer widths jointly by $d = \{d_1, ..., d_{L-1}\}$ in the following corollary.

Corollary 1 (Adjusted from Lee et al. (2019)) *For every test point in $\boldsymbol{x}_* \in \boldsymbol{X}_*$ and $t \geqslant 0$, $f_{lin}(\boldsymbol{x}, \boldsymbol{u}_t)$ converges in distribution as $d \to \infty$ to a Gaussian with mean and covariance given by*

$$\boldsymbol{\mu}(\boldsymbol{X}_*) = \boldsymbol{\Theta}_* \boldsymbol{\Theta}^{-1} \boldsymbol{C}(t) \boldsymbol{Y}, \tag{3.73}$$

$$\begin{aligned} \boldsymbol{\Sigma}(\boldsymbol{X}_*, \boldsymbol{X}_*) =& \boldsymbol{K}_{NNGP**} + \boldsymbol{\Theta}_* \boldsymbol{\Theta}^{-1} \boldsymbol{C}(t) \boldsymbol{K}_{NNGP} \boldsymbol{C}(t) \boldsymbol{\Theta}^{-1} \boldsymbol{\Theta}_* \\ & - (\boldsymbol{\Theta}_* \boldsymbol{\Theta}^{-1} \boldsymbol{C}(t) \boldsymbol{K}_{NNGP*} + h.c.), \end{aligned} \tag{3.74}$$

with matrix $\boldsymbol{C}(t) = (\mathbb{1}_{Nd_y} - \exp(-\eta \boldsymbol{\Theta} t))$. Therefore, over random initialisation, $\lim_{t\to\infty} \lim_{d\to\infty} f_{lin}(x, \boldsymbol{u}_t)$ has a Gaussian distribution with mean and covariance given by Equations (3.73) and (3.74) but with $\boldsymbol{C}(t) = \mathbb{1}_{Nd_y}$ for $\lim_{t\to\infty}$.

This provides a GP for linearised networks in terms of the NTK $\boldsymbol{\Theta}$ and NNGP kernel $\boldsymbol{K}_{\text{NNGP}}$. Note that *h.c.* denotes the Hermitian conjugate. Finally, we connect the infinite-width limit of a standard MLP and its linearisation with the following theorem. Let $\|\cdot\|_F$ denote the Frobenius norm.

Theorem 4 (Adjusted from Lee et al. (2019)) *Let $d = d_1 = \ldots = d_L$ and $\eta_{critical} = 2(\lambda_{min}(\boldsymbol{\Theta}) + \lambda_{max}(\boldsymbol{\Theta}))^{-1}$, where $\lambda_{min/max}$ denotes the min / max eigenvalue of $\boldsymbol{\Theta}$, and assume $\lambda_{min}(\boldsymbol{\Theta}) > 0$. If the learning rate $\eta < \eta_{critical}$, then for all $\boldsymbol{x} \in \mathbb{R}^{d_x}$ with $\|\boldsymbol{x}\|_2 \leqslant 1$, with probability arbitrary close to 1 over random initialisation and for $d \to \infty$*

$$\sup_{t\geqslant 0} \|\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\omega}_t) - \boldsymbol{f}_{lin}(\boldsymbol{x}, \boldsymbol{\omega}_t)\|_2, \ \sup_{t\geqslant 0} \frac{\|\boldsymbol{\omega}_t - \boldsymbol{\omega}_0\|_2}{\sqrt{d}}, \ \sup_{t\geqslant 0} \|\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_0\|_F = \mathcal{O}(d^{-\frac{1}{2}})$$

Moreover, under the same assumptions, $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\omega}_t)$ converges in distribution to the Gaussian given by Equations (3.73) and (3.74) for $d \to \infty$, and to the same Gaussian distribution but with $\boldsymbol{C}(t) = \mathbb{1}_{Nd_y}$ for $d, t \to \infty$.

We refer to Lee et al. (2019) for the complete proof and more details. In summary, Theorem 4 states that, under certain conditions, standard MLPs are linearised networks in the infinite-width limit and that for large width networks, the training dynamics are closely approximated by linearised dynamics. These dynamics involve the time-dependent finite-width NTK $\boldsymbol{\Theta}_t^{(L)}$ which converges to the deterministic NTK $\boldsymbol{\Theta}$ in the infinite-width limit by Theorem 3.[9] For wide networks, the time-dependent NTK $\boldsymbol{\Theta}_t^{(L)}$ exhibits little variance and stays close the finite-width NTK at initialisation $\boldsymbol{\Theta}_0^{(L)}$. Furthermore, Theorem 4 defines an NTK-GP which corresponds to fully-trained neural networks in the infinite-width limit. Performing kernel regression with NTK $\boldsymbol{\Theta}$, as shown in Equation (3.22) and revisited in Equation (3.37), leads to

$$\boldsymbol{f}(\boldsymbol{x}_*) = \boldsymbol{\Theta}_* \boldsymbol{\Theta}^{-1} \boldsymbol{Y} \tag{3.75}$$

which is equivalent to the mean in Equation (3.73) for $t \to \infty$ (and thus $\boldsymbol{C}(t) = \mathbf{1}_{Nd_y}$). Motivated by these results, we study standard and linearised convolutional neural networks at different widths in more detail in Chapter 6, and additionally consider the kernel regression results of the NNGP kernel and NTK in the infinite-width limit.

3.5 DEEP LATENT VARIABLE MODELS

In the previous sections, we showed connections between kernel methods and neural networks. Under particular conditions, we are able to compute

9 Follow up work refines these results and provides bounds on the required widths $d_1, \ldots, d_{L-1}$; see e.g. Arora et al. (2019).

kernels corresponding to infinite-width neural networks which are Gaussian processes. These results are of great theoretical relevance and advance the development of a more rigorous theory of deep learning. However, popular deep learning approaches and finite networks escape these considerations, and rigorous statements about the learned feature mapping $\boldsymbol{\phi}$ are challenging due to the non-linear and sequential concatenation of different layers. In practice, we make prior assumptions, as for example on the type of architecture, which can be viewed as inductive biases steering learning and generalisation. A popular and broader type are encoder-decoder architectures. In contrast to the idea of lifting the data into a higher-dimensional feature representation as presented so far, these architectures typically encode inputs into a lower-dimensional representation which captures the relevant information necessary to decode the output of interest. For the remainder of this chapter, we focus on the guiding idea of compression and start by considering latent variable models in the context of deep learning.

3.5.1 *Latent Variable Models*

Latent variable models (LVMs) are statistical models which differentiate between observable or manifest random variables X or Y and latent random variables Z which are unobserved or hidden. We denote the realisations or observed values of these random variables by $\boldsymbol{x} \in \mathbb{R}^{d_x}$, $\boldsymbol{y} \in \mathbb{R}^{d_y}$, and latent values by $\boldsymbol{z} \in \mathbb{R}^{d_z}$. As before, let $\mathcal{D} = \{(\boldsymbol{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$ denote our (training) dataset. Typically, latent variables Z serve as a bottleneck with $d_z < d_x$ and thus as a compressed representation of X with likelihood $p(X = \boldsymbol{x}_n \mid Z = \boldsymbol{z}_n)$ and prior $p(Z = \boldsymbol{z}_n)$ for a data point $n \in \{1, ..., N\}$.[10] Hence, LVMs describe a generative process for the observed data $\mathcal{D}$. Depending on the form of the likelihood and the prior, different LVMs are defined. Typical examples include (i) mixture models like the mixture of Gaussians with discrete latent variables for different cluster assignments, (ii) factor analysis with continuous variables modelling observed variables as linear combinations of latent factors of which (iii) principal component analysis is a particular special case, (iv) canonical correlation analysis with shared or correlated latent variables for observed variables X and Y as well as independent latent variables, or (v) independent component analysis. However, all of these LVMs constitute models in which linear relationships between observed and latent variables are assumed.

10 To stay consistent with the notation in the previous sections, we denote probability densities by p and adopt the convention of dropping the random variable, i.e. $p(X = \boldsymbol{x}) \equiv p(\boldsymbol{x})$.

In the context of deep learning, deep generative models typically represent non-linear (deep) latent variable models. Deep generative models (DGMs) are a broad class of (usually) unsupervised machine learning approaches. A common feature of these neural network approaches is that they attempt to approximate high-dimensional probability distributions based on a large number of samples which enables likelihood estimation and generating of novel objects following the underlying distribution (Ruthotto and Haber, 2021). Popular DGMs include variational autoencoders (VAEs) (Kingma and Welling, 2014; Rezende et al., 2014), generative adversarial networks (GANs) (Goodfellow et al., 2020), normalising flows (Rezende and Mohamed, 2015), and recently diffusion models (Sohl-Dickstein et al., 2015; Song and Ermon, 2019; Ho et al., 2020). In the following, we focus on the VAE and extensions of bottleneck models.

3.5.2 *Variational Autoencoder*

The overarching goal of generative modelling is to learn a latent representation of the intractable probability distribution $p(\boldsymbol{x})$ of the input data points $\boldsymbol{x}$. For this purpose, we devise a deterministic generator or decoder $\boldsymbol{g}$: $\mathbb{R}^{d_z} \rightarrow \mathbb{R}^{d_y}$ which allows us to map samples of a tractable latent distribution $p(\boldsymbol{z})$ into the input domain such that $\boldsymbol{g}(\boldsymbol{z}) \approx \boldsymbol{x}$ generates samples in input space. We can approximate decoder $\boldsymbol{g}$ with a neural network parametrised by $\boldsymbol{\theta}$. By using the likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ for decoder $\boldsymbol{g}_{\boldsymbol{\theta}}(\boldsymbol{z})$ and the prior $p(\boldsymbol{z})$, we cast the task as computing the marginal likelihood or evidence

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})d\boldsymbol{z}. \tag{3.76}$$

The evaluation of the high-dimensional integral, however, is intractable in general. Using Bayes' theorem, we can write the marginal likelihood as

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}. \tag{3.77}$$

Maximising the marginal likelihood with respect to $\boldsymbol{\theta}$ is again infeasible. While we can define a model for the likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ and choose a prior $p(\boldsymbol{z})$, the posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ is generally intractable. In case of real-valued data, a typical choice for the likelihood is the Gaussian distribution

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = (2\pi\sigma_{\text{dec}}^2)^{-d_x/2} \exp\left(-\frac{1}{2}\sigma_{\text{dec}}^{-2}\|\boldsymbol{g}_{\boldsymbol{\theta}}(\boldsymbol{z}) - \boldsymbol{x}\|_2^2\right) \tag{3.78}$$

where σ^2_{dec} denotes the decoder noise variance. The negative log-likelihood of this equation corresponds to the MSE loss, which we have previously derived in Equation (3.12). A key feature of VAEs is to approximate the posterior $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ with a family of tractable probability distributions through a variational inference approach. To this end, we introduce a (inference) neural network, the encoder, parametrised by $\boldsymbol{\phi}$ which defines the approximate (variational) posterior

$$q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \approx p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}). \tag{3.79}$$

A multivariate Gaussian distribution with $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}), \text{diag}(\boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\boldsymbol{x})))$ is a typical choice for $q_{\boldsymbol{\phi}}$. Thus, the encoder learns the mean $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x})$ and variance $\boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\boldsymbol{x})$[11] for different inputs $\boldsymbol{x}$ while using the same set of parameters $\boldsymbol{\phi}$, which is referred to as amortised inference. We provide a schematic illustration of the model in Figure 3.4 where the upper branch corresponds to the underlying idea of the VAE. Note that the dashed line in the figure indicates that the latent representations $\boldsymbol{z}$ are obtained as samples from $\mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}), \text{diag}(\sigma^2_{\boldsymbol{\phi}}(\boldsymbol{x})))$. In order to quantify the error in the approximation, we can make use of the statistical distance defined in the following.

Definition 5 (Kullback-Leibler Divergence) *Consider two probability distributions P and Q defined on the same probability space $\mathcal{X}$. Then the relative entropy or Kullback-Leibler divergence is defined as*

$$D_{KL}\left(P \| Q\right) = \int p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} d\boldsymbol{x} \tag{3.80}$$

if P and Q are distributions of continuous random variables with p and q denoting the probability densities, respectively, and

$$D_{KL}\left(P \| Q\right) = \sum_{\boldsymbol{x} \in \mathcal{X}} P(\boldsymbol{x}) \log \frac{P(\boldsymbol{x})}{Q(\boldsymbol{x})} \tag{3.81}$$

if P and Q are discrete probability distributions.

Note that by this definition, the Kullback-Leibler (KL) divergence is (i) non-symmetric with $D_{\text{KL}}\left(P \| Q\right) \neq D_{\text{KL}}\left(Q \| P\right)$ and (ii) non-negative with

11 For numerical reasons, we learn the log-variance $\log(\sigma^2_{\boldsymbol{\phi}}(\boldsymbol{x}))$ in practice.

$D_{\mathrm{KL}}\left(P \| Q\right) \geqslant 0$ and equals zero if and only if $P \equiv Q$. With this in place, we can derive a tractable expression for the marginal log-likelihood

$$
\begin{aligned}
\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \mathbb{E}_{q_{\boldsymbol{\phi}}(z|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x})] && (3.82)\\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(z|\boldsymbol{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, z)}{q_{\boldsymbol{\phi}}(z|\boldsymbol{x})}\right] + \mathbb{E}_{q_{\boldsymbol{\phi}}(z|\boldsymbol{x})}\left[\log \frac{q_{\boldsymbol{\phi}}(z|\boldsymbol{x})}{p_{\boldsymbol{\theta}}(z|\boldsymbol{x})}\right] && (3.83)\\
&= \mathrm{ELBO}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\boldsymbol{x}) + D_{\mathrm{KL}}\left(q_{\boldsymbol{\phi}}(z|\boldsymbol{x}) \| p_{\boldsymbol{\theta}}(z|\boldsymbol{x})\right) && (3.84)\\
&\geqslant \mathrm{ELBO}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\boldsymbol{x}) && (3.85)
\end{aligned}
$$

where we identify two important terms in Equation (3.83). The second term is the KL divergence between the approximate and true posterior $q_{\boldsymbol{\phi}}(z|\boldsymbol{x})$ and $p_{\boldsymbol{\theta}}(z|\boldsymbol{x})$ which is non-negative. Thus, we can view the first term as a variational lower bound to the marginal log-likelihood, known as the evidence lower bound (ELBO). Note that the second term not only provides a measure of distance between approximate and true posterior, but also quantifies the gap or tightness of the ELBO and marginal log-likelihood. For the ELBO we can derive

$$
\begin{aligned}
\mathrm{ELBO}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\boldsymbol{x}) &= \mathbb{E}_{q_{\boldsymbol{\phi}}(z|\boldsymbol{x})}\left[\log(p_{\boldsymbol{\theta}}(\boldsymbol{x}, z)) - \log(q_{\boldsymbol{\phi}}(z|\boldsymbol{x}))\right] && (3.86)\\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(z|\boldsymbol{x})}\left[\log(p_{\boldsymbol{\theta}}(\boldsymbol{x}|z)) + \log p(z) - \log(q_{\boldsymbol{\phi}}(z|\boldsymbol{x}))\right] && (3.87)\\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}(z|\boldsymbol{x})}\left[\log(p_{\boldsymbol{\theta}}(\boldsymbol{x}|z))\right] - D_{\mathrm{KL}}\left(q_{\boldsymbol{\phi}}(z|\boldsymbol{x}) \| p(z)\right) && (3.88)
\end{aligned}
$$

where the first term in the expectation of Equation (3.88) is the log-likelihood and the second term is KL divergence between prior and approximate posterior. Note that the goal is to maximise the ELBO. Casting this into a minimisation problem, we obtain the objective function of the VAE as

$$
\begin{aligned}
J(\boldsymbol{\phi}, \boldsymbol{\theta}) &= \mathbb{E}_{p(\boldsymbol{x})}\left[\mathbb{E}_{q_{\boldsymbol{\phi}}(z|\boldsymbol{x})}\left[-\log(p_{\boldsymbol{\theta}}(\boldsymbol{x}|z))\right] + D_{\mathrm{KL}}\left(q_{\boldsymbol{\phi}}(z|\boldsymbol{x}) \| p(z)\right)\right] && (3.89)\\
&\approx \frac{1}{s}\sum_{i=1}^{s} \mathbb{E}_{q_{\boldsymbol{\phi}}(z|\boldsymbol{x}_i)}\left[-\log(p_{\boldsymbol{\theta}}(\boldsymbol{x}_i|z))\right] + D_{\mathrm{KL}}\left(q_{\boldsymbol{\phi}}(z|\boldsymbol{x}_i) \| p(z)\right). && (3.90)
\end{aligned}
$$

In practice, we estimate the expectation with respect to the intractable distribution $p(\boldsymbol{x})$ with Monte Carlo samples, i.e. we compute the mean with respect to s data samples which typically is a mini-batch of the input data. The first term in the expectation of Equation (3.90) is the negative log-likelihood which can be interpreted as a reconstruction error of the input $\boldsymbol{x}$. Considering a Gaussian decoder as given in Equation (3.78) and its negative log-likelihood in Equation (3.12) motivates the use of the MSE loss for the reconstruction loss. The second term in the expectation of Equation (3.90) can interpreted as a regulariser which biases the approximate posterior towards the prior distribution of latent variables $\boldsymbol{z}$. A typical choice for the prior is a standard normal distribution $\mathcal{N}(0, \mathbb{1}_{d_z})$. Having both prior and approximate posterior as Gaussian distributions enables a closed-form expression for the KL divergence. However, the objective function of the VAE reveals a potential conflict between reconstruction fidelity and the regularisation towards a standard normal distribution, which cannot be fulfilled simultaneously. Two viable approaches are to introduce a Lagrange hyperparameter β which scales the regularisation term relative to the reconstruction error, as pursued in the β-VAE (Higgins et al., 2017), or to balance the reconstruction error through the decoder noise σ^2_{dec} in case of a Gaussian decoder. Another popular choice for the decoder likelihood is the Bernoulli distribution with

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \prod_{i=1}^{d_x} g_{\boldsymbol{\theta}}(\boldsymbol{z})_i^{x_i} \left(1 - g_{\boldsymbol{\theta}}(\boldsymbol{z})_i\right)^{(1-x_i)} \tag{3.91}$$

whose negative log-likelihood gives rise to the (binary) cross-entropy loss and thus is particularly suitable for binary data.

The encoder and decoder networks can be implemented by any kind of architecture suitable for processing the input data, like MLPs, convolutional neural networks, or graph neural networks. In order to learn the encoder parameters $\boldsymbol{\phi}$ with a derivative-based minimisation, the reparametrisation trick (Kingma and Welling, 2014) is used where the latent sample $\boldsymbol{z} \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ can be written as function of noise variable $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbb{1}_{d_z})$ as

$$\boldsymbol{z}(\boldsymbol{\epsilon}) = \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}) + \operatorname{diag}\left(\boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\boldsymbol{x})\right) \boldsymbol{\epsilon}. \tag{3.92}$$

3.5.3 *Deep Variational Information Bottleneck*

An information theoretic approach to compression models is the Information Bottleneck (IB) principle (Tishby et al., 1999). Consider the following setting: For a random variable X corresponding to a signal we want to identify a compression given by a random variable Z which maintains meaningful or relevant information about a relevance random variable Y,

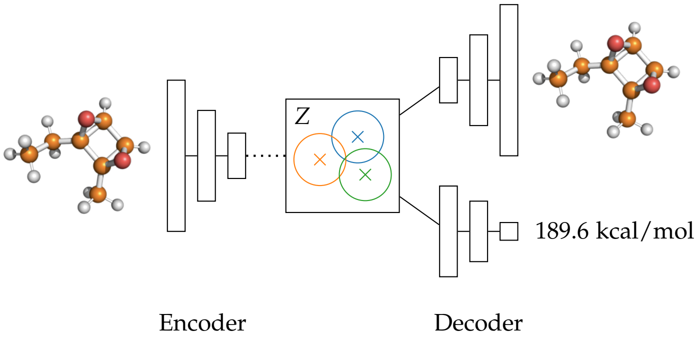


Figure 3.4: Schematic illustration of a hybrid model combining VAE and DVIB decoder branches as used in Chapter 5. The encoder processes the input $\boldsymbol{x}$, e.g. a representation of a molecule, and parametrises the mean $\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x})$ (depicted by crosses $\times$) and variance $\boldsymbol{\sigma}^2_{\boldsymbol{\phi}}(\boldsymbol{x})$ (depicted by circles) used to map $\boldsymbol{x}$ into lower-dimensional latent space Z (different colours depict different mappings of $\boldsymbol{x}$). These latent codes are decoded by two different decoder branches: The upper branch corresponds to the standard VAE setting and reconstructs input $\boldsymbol{x}$, e.g. the molecule representation. The lower branch corresponds to the DVIB setting and predicts a property $\boldsymbol{y}$, e.g. the band gap energy of the molecule.

satisfying the Markov chain $Z \leftrightarrow X \leftrightarrow Y$. In order to formulate the IB problem, we make use of the notion of mutual information.

Definition 6 (Mutual Information) *Let X and Y be two random variables with joint distribution $P(X,Y)$ and marginal distributions $P(X)$ and $P(Y)$. The mutual information is defined as*

$$I(X;Y) = D_{KL}\left(P(X,Y) \| P(X)P(Y)\right). \tag{3.93}$$

Informally, the mutual information (MI) measures the amount of information we obtain about one random variable by observing the other random variable. Note that this definition implies that the MI is (i) symmetric with $I(X;Y) = I(Y;X)$ and (ii) non-negative with $I(X;Y) \geqslant 0$. The IB variational problem is then defined as

$$\min_{P(Z|X)} I(X;Z) - \lambda I(Z;Y). \tag{3.94}$$

A low MI in the first term $I(X;Z)$ corresponds to a high level of compression, while a low MI in the second term $I(Z;Y)$ represents that little relevant information about Y is preserved in Z. Thus, parameter λ controls the trade-off between the compression of X and preservation of relevant information about Y. The solution provides the optimal conditional distribution of Z given X. Generally, no analytical solutions exists but numerical approximations can be used (Tishby et al., 1999). For the special case of a joint Gaussian distribution,

$$(X, Y) \sim \mathcal{N}\left(0, \begin{pmatrix} \Sigma_X & \Sigma_{XY} \\ \Sigma_{XY}^\top & \Sigma_Y \end{pmatrix}\right) \tag{3.95}$$

provides the Gaussian information bottleneck for which the solution to Equation (3.94) is also a Gaussian distribution. The random variable Z can be represented through the linear transformation $Z = AX + \epsilon$, with $\epsilon \sim \mathcal{N}(0, \Sigma_\epsilon)$ and projection matrix A, and thus is distributed according to $\mathcal{N}(0, \Sigma_Z)$ with $\Sigma_Z = A\Sigma_X A^\top + \Sigma_\epsilon$. For the optimal Z, we obtain $\Sigma_\epsilon = \mathbb{1}$ and the mutual information between X and Z is then equal to

$$I(X; Z) = \tfrac{1}{2} \log |A\Sigma_X A^\top + \mathbb{1}|, \tag{3.96}$$

where $|\cdot|$ denotes the determinant of the matrix. For further details on the Gaussian IB we refer to Chechik et al. (2005). By additionally assuming that the projection matrix A is diagonal, we obtain a sparse compression Z (Rey et al., 2014). This can be motivated by observing that a full-rank projection AX' of X' does not change the mutual information $I(X; X') = I(X; AX')$. A reduction in mutual information can only be achieved by a rank-deficient matrix A, i.e. by setting diagonal elements to zero.

The deep variational information bottleneck (DVIB) is a variational approximation to Equation (3.94) based on neural networks (Alemi et al., 2017). For this, we rewrite the objective function as

$$J(\boldsymbol{\phi}, \boldsymbol{\theta}) = I_{\boldsymbol{\phi}}(X; Z) - \lambda I_{\boldsymbol{\phi},\boldsymbol{\theta}}(Z; Y) \tag{3.97}$$

where the conditional distributions $p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ and $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{z})$ are parametrised by neural networks and represent the encoder and decoder, respectively. This is similar to the VAE setting in the previous section, but for a supervised problem. A schematic illustration of the model is shown in Figure 3.4 where the lower branch corresponds to the underlying idea of the DVIB. Following Wieczorek and Roth (2020), we can derive

$$I_{\boldsymbol{\phi}}(X;Z) = D_{\mathrm{KL}}\left(p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})p(\boldsymbol{x})\|p(\boldsymbol{z})p(\boldsymbol{x})\right) \tag{3.98}$$

$$= \mathbb{E}_{p(\boldsymbol{x})}\left[D_{\mathrm{KL}}\left(p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z})\right)\right] \tag{3.99}$$

and

$$I_{\boldsymbol{\phi},\boldsymbol{\theta}}(Z;Y) = D_{\mathrm{KL}}\left(\left(\int p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})p(\boldsymbol{x},\boldsymbol{y})\|p(\boldsymbol{z})p(\boldsymbol{y})\right)\right) \tag{3.100}$$

$$\geqslant \mathbb{E}_{p(\boldsymbol{x},\boldsymbol{y})}\left[\mathbb{E}_{p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{z})\right]\right] - \mathbb{E}_{p(\boldsymbol{y})}\left[\log p(\boldsymbol{y})\right] \tag{3.101}$$

where the last term in Equation (3.101) is the (differential) entropy. Note that in addition to Markov chain $Z \leftrightarrow X \leftrightarrow Y$ also $X \leftrightarrow Z \leftrightarrow Y$ holds for the DVIB by construction. Wieczorek and Roth (2020) showed that only the latter Markov chain is required which leads to the lower bound provided in Equation (3.101). A thorough analysis of the differences between the IB and DVIB approach is provided in Wieczorek and Roth (2020). Comparing the DVIB objective function given by Equations (3.97), (3.99), and (3.101) with the VAE objective function given in Equation (3.90) shows a correspondence between these models. Their connection is discussed in Alemi et al. (2017) in more detail.

In Chapter 5 we make use of a hybrid approach which combines both decoders of the VAE and DVIB in a unifying framework, as depicted in Figure 3.4. The latent representation $\boldsymbol{z}$ is used for both the reconstruction of the initial input $\boldsymbol{x}$ as well as for the prediction of a continuous target $\boldsymbol{y}$. In the hybrid model, we adopt the presented VAE approach and use the approximate posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$ as well as Gaussian decoders and add the additional loss in Equation (3.101) to the model objective. Similar models were used in our work on deep archetypal analysis (Keller et al., 2019, 2021), in the symmetry-transformation information bottleneck (STIB) (Wieser et al., 2020) or the chemical VAE (Gómez-Bombarelli et al., 2018).

3.6 SEMANTIC SEGMENTATION

An important class of neural network architectures are convolutional neural networks (CNNs). The guiding idea behind this type of network is the prior assumption that local relationships in the data, like correlation of adjacent pixels in an image, are relevant in certain tasks, e.g. recognition of objects in images. Typical applications are classification, object detection, or recognition tasks in images, but also other applications which involve data arranged in matrices or sequences. The first architectures of that kind were proposed in the 1980s (Fukushima, 1980; LeCun et al., 1989) and were inspired by biological hierarchical models of the primary visual cortex (Hubel and Wiesel, 1959) and their resemblance to the convolution operation in mathematics. A CNN can be regarded as a particular kind of MLP, where certain connections between units of different layers in Figure 3.2 are missing and some connections share the same parameters. This defines convolutional layers which employ filters that apply the convolution operation to map an input image to a feature map. By concatenating several such layers, convolutional filters map between feature maps and can learn increasingly complex features. The filters can be thought of as randomly initialised matrices, often of size 3×3, which stride over the input or feature map and store the sum of all element-wise products in the resulting feature map. During training, these filters are updated in order to extract meaningful features. In addition, pooling layers are used to down-sample feature maps. Reducing the resolution incorporates a local invariance to small shifts of objects in the input image. A common choice are max pooling layers which take the maximum of 2×2 adjacent pixels. The last part of a CNN typically consists of fully-connected layers similar to the MLP. Figure 3.5 illustrates such a CNN known as LeNet-5 (LeCun et al., 1998), which can be viewed as the foundation of many CNNs proposed in the last years. The

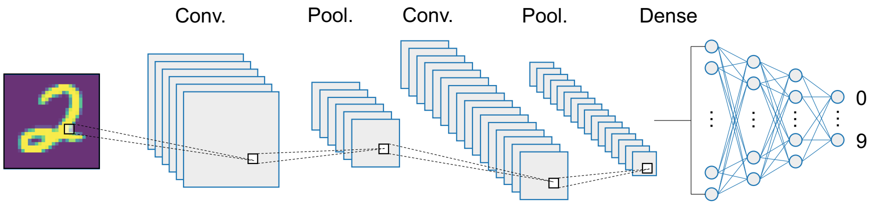


Figure 3.5: Illustration of a convolutional neural network. The different layers consist of convolutional layers, pooling layers, and fully-connected (dense) layers. Feature maps and hidden units are depicted at the different layers.

recent popularity of CNNs was sparked with AlexNet (Krizhevsky et al., 2012) and a great variety of other CNN *families* followed, like ResNets (He et al., 2016), R-CNNs (Girshick et al., 2014), YOLO models (Redmon et al., 2016), and many others. Recently, transformer approaches (Vaswani et al., 2017) like the vision transformer (Dosovitskiy et al., 2021) challenge the dominance of CNNs in vision applications. In Chapter 6, we focus on LeNet and AlexNet and study these CNNs as well as their linearised versions at different network widths and for increasingly difficult classification tasks.

Apart from classification tasks, CNN approaches are particularly useful in semantic segmentation. An example for such a segmentation task is illustrated in Figure 3.6 on the left. An aerial image containing a shallow landslide (see Chapter 2) is shown. The provided segmentation mask highlights the relevant pixels belonging to the erosion classes. In Chapter 4 we present a CNN approach for efficient and scalable segmentation of such erosion sites based on the U-Net (Ronneberger et al., 2015), which has a particular focus in this thesis. The U-Net is a fully-convolutional network

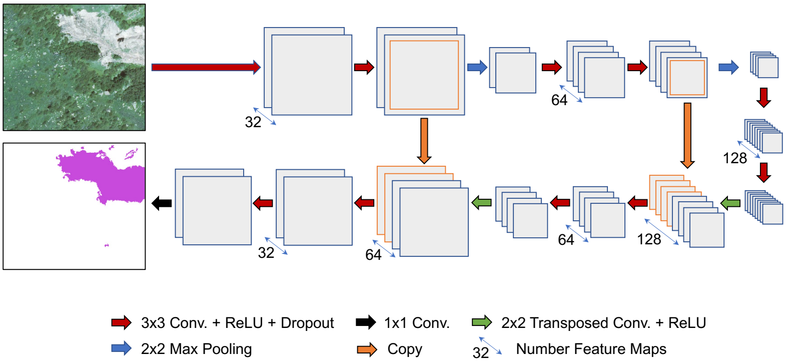


Figure 3.6: Semantic segmentation with the U-Net. On the left an input image (top) and the segmentation result (bottom) with highlighted region of interest (in magenta) are shown. The U-Net can be viewed as an encoder-decoder architecture. In the upper part, a compressed representation of the input is learned. In the lower part, the compressed representation is decoded into a segmentation map. Skip connections allow high-frequency information to pass from the encoder to the decoder.

(Long et al., 2015) and can be viewed as a multi-scale encoder-decoder architecture, as illustrated in Figure 3.6. The main components are convolution, max pooling, dropout, and transposed convolution operations with rectified linear unit (ReLU) activations $\sigma(\cdot) = \max\{0, \cdot\}$. The transposed convolution operation[12] is used to up-sample single pixels from a feature map to typically 2×2 pixels in the subsequent feature map. Thus, it can be viewed as an inverse operation to max pooling. Lastly, dropout is employed a regularisation approach which sets a subset of unit activations to zero.

12 Sometimes referred to as up-convolution, fractionally-strided convolution, or deconvolution.

For instance, with a dropout probability of 50% about half of the units are randomly set to zero for a single training iteration. This usually improves the stability and accuracy of prediction outcomes.

In the encoder part (top layers in Figure 3.6), a compressed representation of the input is learned by sequences of two convolutional layers with ReLU activations followed by a max pooling layer. The bottleneck of this architecture is given by the feature maps before the first transposed convolution operation (bottom right in Figure 3.6). Note that in the depicted U-Net with two max pooling steps, one pixel in the bottleneck feature maps is the result of 32×32 processed pixels in the input image which is referred to as the receptive field. In the decoder part (bottom layers in Figure 3.6), the compressed representation is expanded to the original image size[13] by sequences of transposed convolutional layers with ReLU activations followed by two convolutional layers and ReLU activations. An important aspect of the U-Net is the usage of skip connections, in which feature maps from the encoder part are appended to the feature maps obtained through the transposed convolutions to provide high-frequency details in the decoder part. Finally, a 1×1 convolutional layer performs a convolution only on the channel dimension and provides the segmentation maps for the individual classes. In the last step, a pixel-wise softmax activation function $\sigma(z)_i = \exp(z_i)/\sum_{j=1}^{d_z} \exp(z_j)$, $i, j \in \{1, ..., d_z\}$, rescales the activations for each pixel to the $[0, 1]$ interval. We cover more application relevant details in Chapter 4.

The U-Net has experienced a great popularity in semantic segmentation tasks, due to its relative architectural simplicity and applicability in domains with small training sets, in particular in biomedical but also in geoscience applications. A great number of extensions was proposed leading to a large *zoo* of different U-Net architectures. Prominent examples include the 3D-U-Net (Çiçek et al., 2016), U-Nets with additional skip-connectivity (Zhou et al., 2018; Huang et al., 2020a), or the unifying framework of the nnU-Net (Isensee et al., 2021), but also many more. U-Nets are also used for inverse problems (Jin et al., 2017; Han and Ye, 2018). Building on the success of models employing the attention mechanism (Vaswani et al., 2017), recent extensions incorporate attention into U-Nets like (Oktay et al., 2018) and find applications in geoscience as for deforestation detection (John and Zhang, 2022). Furthermore, U-Net-like architectures (Hatamizadeh et al.,

13 Note that the output size depends on the padding used in the convolutional layers. If no padding is used, the output size is smaller than the input image size as pixels on the borders are missing. With padding, the original image size is retrieved.

2022; Chen et al., 2021) based on the vision transformer were proposed. An additional direction develops probabilistic approaches to semantic segmentation like the Bayesian U-Net (Dechesne et al., 2021) or the (hierarchical) probabilistic U-Net (Kohl et al., 2018, 2019) for segmentation in presence of label ambiguity. We review these approaches as potential extensions of our work in Chapter 7.

4 SEMANTIC SEGMENTATION OF EROSION PHENOMENA

The intactness of soil is of great relevance for plants, animals, humans, and the ecosystem as a whole. Soil erosion in alpine grasslands poses a major threat to ecosystem services of alpine soils. Natural causes for the occurrence of soil erosion are steep topography and prevailing climate conditions in combination with soil fragility. To increase our understanding of ongoing erosion processes and support sustainable land-use management, there is a need to acquire detailed information on spatial occurrence and temporal trends. Existing approaches to identify these trends are typically laborious, lack transferability to other regions, and are consequently only applicable to smaller regions. In order to overcome these limitations and create a sophisticated erosion monitoring tool capable of large-scale analysis, we develop a model based on the U-Net, a fully-convolutional neural network, to map different erosion processes on high-resolution aerial images (RGB, $0.25 - 0.5$ m pixel resolution).

In a first study, we train a U-Net on a high-quality dataset consisting of labelled erosion sites mapped with object-based image analysis (OBIA) in the Urseren valley (Central Swiss Alps) for five aerial images (16-year period). We use the U-Net model to map the same study area and conduct quality assessments based on a held-out test region and a temporal transferability test on new images. Erosion classes are assigned according to their type (shallow landslide and sites with reduced vegetation affected by sheet erosion) or land-use impacts (livestock trails and larger management affected areas). We show that results obtained through OBIA and the U-Net follow similar linear trends for the 16-year study period, exhibiting increases in total degraded area of 167% and 201%, respectively. Segmentations of eroded sites are generally in good agreement, but also display method-specific differences, which lead to an overall precision of 73%, a recall of 84%, and an F_1-score of 78%.

In a second study, we extend the U-Net training to eight different grassland valleys all over Switzerland and evaluated on the same test region in the Urseren valley as well as another spatially separated region, the

Part of this chapter has been published in Samarin et al. (2020). The content related to the publication was covered in the dissertation of Zweifel (2021), too.

Turbach valley. Incorporating multiple valleys with diverse conditions, we show an improvement of segmentation results which generally leads to more sites being predicted as degraded area and retrieves 94% of the U-Net segmentation sites of the first study. Furthermore, similar linear trends for the 16-year study period in the Urseren valley are obtained with a relative increase of 132% of total degraded area in the U-Net prediction as compared to the 167% in the OBIA baseline.

Our results show that the U-Net approach is transferable to spatially and temporally unseen data, i.e. to regions within the Urseren valley but also beyond (as shown for the Turbach valley) and data from new years. Therefore, it is a suitable method to efficiently and successfully capture the temporal trends and spatial heterogeneity of degradation in alpine grasslands. Additionally, the U-Net is a powerful and robust tool to map erosion sites in a predictive manner utilising large amounts of new aerial imagery.

4.1 MONITORING SOIL DEGRADATION

As introduced in Chapter 2, soil degradation is a major ecological threat which affects many areas of the world and can be accelerated by land-use management and changing climate parameters, such as precipitation and temperature (EEA, 2009; Fuhrer et al., 2006; Meusburger and Alewell, 2008; Nearing et al., 2004; Scheurer et al., 2009). In Switzerland, some alpine grassland areas are strongly affected by soil erosion due to the steep terrain and extreme climate conditions. While soil erosion occurs naturally in these environments – in the form of landslides (triggered by snow gliding or heavy precipitation events) or sheet erosion (the process of the removal of topsoil caused by rain drops' impacts and overland flow) – there are also anthropogenic influences (e.g. agricultural activities) which can accelerate erosion rates (Meusburger and Alewell, 2008; Tasser et al., 2003; Zweifel et al., 2019). For example, livestock keeping can lead to overgrazing and trampling in favoured grazing areas. Over time, livestock trails develop and trampling and grazing can lead to a reduction in vegetation cover, which in turn is prone to sheet erosion (Apollo et al., 2018; Torresani et al., 2019). Additionally, livestock keeping can cause instabilities on slopes and ultimately result in landslides (Wiegand and Geitner, 2010). We provide visual illustrations of these erosion phenomena in Figures 2.1 and 4.3.

Therefore, erosion processes have strong temporal and spatial dynamic components, which is why large-scale understanding and detailed mapping over time and space is of great importance for long-term sustainable management practices.

Alpine areas are difficult to access and erosion features can affect substantial areas, making a comprehensive understanding of ongoing erosion processes unattainable from the ground. Larger-scale erosion studies for Switzerland have mainly been approached with the help of soil erosion modelling – e.g. the (revised) universal soil loss equation (Alder et al., 2015; Bircher et al., 2019; Meusburger et al., 2010, 2012; Prasuhn et al., 2013; Schmidt et al., 2018, 2019a,b). To achieve a thorough understanding of potential soil erosion threats, it is important to combine model outputs with observations for validation purposes (Fischer et al., 2018). The latter is especially crucial in mountainous and grassland areas, where model suitability has been questioned (see discussion in Alewell et al. (2019)). High-resolution aerial imagery offers the opportunity to remotely assess and map the spatial extent of bare soil sites and sites with strongly reduced vegetation cover, allowing certain constraints to be overcome, such as the inaccessibility or extent of a study area. Object-based image analysis (OBIA) is an approach commonly used to identify urban and natural "objects" on satellite and aerial imagery and has been successfully used in the past to map various forms of soil erosion (D'Oleire-Oltmanns et al., 2012; Eisank et al., 2014; Guzzetti et al., 2012; Hölbling et al., 2015, 2016, 2020; Martha et al., 2012; Shruthi et al., 2011; Wang et al., 2020; Wiegand et al., 2013; Zweifel et al., 2019). OBIA creates image segments by grouping pixels with similar properties together, which can then be classified based on object information (spectral, spatial, textural, and contextual) with expertly developed classification rules including various machine learning classifiers. OBIA is a method suitable for smaller study areas, but large-scale studies become difficult to manage. Limitations including processing times, a lack of work-flow transferability to other scenes, and the involvement of manual steps hinder efficient spatial up-scaling of projects. In past years, deep learning methods have progressively been applied in the field of remote sensing for image classification tasks and segmentation tasks (Ma et al., 2019; Heydari and Mountrakis, 2019; Huang et al., 2018; Yuan et al., 2020). In this study, we apply a deep learning method to demonstrate that it is capable of mapping and classifying soil erosion features on aerial images in a fast, objective, reliable, and scalable manner. We apply a fully-convolutional neural network (CNN) framework using the U-Net architecture developed by Ronneberger et al. (2015). In general, the U-Net architecture offers itself to semantic segmentation tasks with limited training data. The U-Net and variations of this architecture have become increasingly popular for remote sensing tasks. Many applications focus on urban settings for road (Yuan et al., 2019; Zhang et al., 2018c; Alshaikhli et al., 2019; Wulamu et al., 2019) or building extraction (Xu et al., 2018; Yi et al., 2019; Ivanovsky et al., 2019; Mboga et al., 2019) from satellite and aerial imagery. Applications in a natural environment are constrained by the limited availability of high-quality labelled training data. Despite this limitation, the U-Net has been applied in cloud detection on satellite images (Yang et al., 2019), mapping of woody vegetation (Flood et al., 2019), segmentation of plant species (Kattenborn et al., 2019), forest damage assessment (Hamdi et al., 2019), the extraction of Antarctic glacier and ice shelf fronts (Baumhoer et al., 2019), and archaeological studies (Bundzel et al., 2020), to name a few. Our annotated training data has been generated by mapping erosion sites on aerial images using OBIA for a valley in the Central Swiss Alps (Urseren valley, Canton of Uri). We compare the U-Net results to OBIA mapping for a held-out test region (area of 17 km^2), which was not used for training (9 km^2) for the years 2000, 2004, 2010, and 2013. Additionally, we investigate both the temporal and the spatial transferability of the U-Net method by mapping a new test aerial image not seen during training (2016). Our main objectives of this study are twofold: firstly, to show that the fully automated U-Net approach is capable of reproducing the high-quality soil erosion mapping and the temporal trends as they were attained with OBIA for the same study site; secondly, to show that the U-Net approach generalises well to new aerial images, i.e. can be used in a predictive manner to perform adequate segmentation of previously unseen input data. In contrast, the OBIA procedure typically eludes such predictive usage and needs to be adjusted for each new aerial image. The capabilities and the fully automated nature of the U-Net approach make it a highly promising tool for efficient large-scale erosion mapping, e.g. alpine-wide analysis of soil erosion in semi-natural ecosystems such as grasslands and bush-land.

4.2 CASE STUDY: URSEREN VALLEY

The Urseren valley (26 km^2) is an alpine valley located in Central Switzerland in the southern part of the Canton of Uri, as illustrated in Figure 4.1. The valley has a NE/SW orientation, and exhibits steep slopes (average angle of 27°) and rough terrain. The valley is geologically divided into two distinct sections and separated by the river Reuss: The northern slope is part of the Aarmassif (granite), and the southern slope belongs to the Goatherd massif (gneiss). Located between these two massifs near the valley floor is the so-called Urseren-Garvera zone (Mesozoic sediments) (Wyss, 1986). The dominant soil types in the catchment are Podzols and Cambisols, with Leptosols commonly found on steep slopes (classified after IUSS Working Group WRB (2006)).

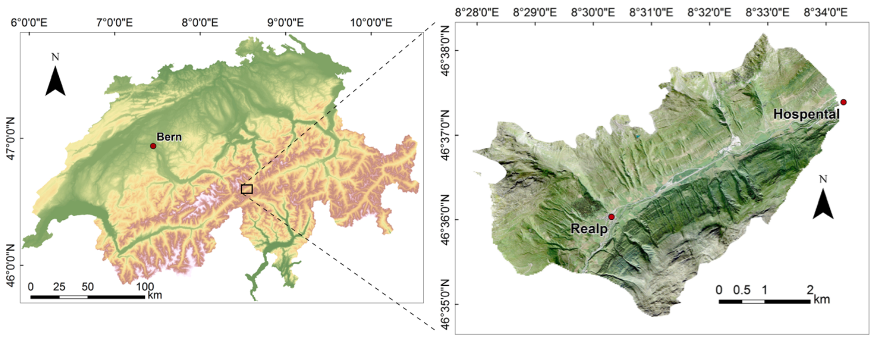


Figure 4.1: The Urseren valley is located in the Central Swiss Alps in the Canton of Uri. The left map contains the topographic map of Switzerland (from low elevations in green to high elevations in brown to white). The right image contains an aerial image of the Urseren valley overlaid on a hill-shade map of the area.

The 30-year average temperature (1990 – 2019) of the closest meteorological station in Andermatt (1438 m a.s.l.) is 3.9°C. The average temperature has increased by 0.7°C during the last 10 years (compared to the average of 1980 – 2009). The average rainfall during the last 30 years was 1384 mm with an average maximum 3-day precipitation intensity of 123 mm/3 d. The average seasonal (November–April) snow height is 58 cm with maximum snow heights during February/March (average of 103 cm) (data provided by MeteoSwiss, 2020). The dominant land-covers are grassland (including dwarf-shrubs consisting of Calluna vulgaris, Rhododendron ferrugineum, and Juniperus sibirica), which is mainly used for grazing (i.e. sheep and cattle) and haying, shrubs (mainly Alnus viridis and Sorbus aucuparia), and debris/bare rock areas (Meusburger and Alewell, 2008). Shrub encroachment due to land abandonment and extensification is present in the valley. Avalanches and snow gliding occur frequently in the Urseren valley, facilitated by the deforested state of the slopes. The dominant erosion processes in this region are (shallow) landslides, sheet erosion, and erosion caused by land-use management (livestock, machinery, and manuring). Additional information on the Urseren valley and occurring erosion processes can be found in Alewell et al. (2015); Meusburger and Alewell (2008); Zweifel et al. (2019).

4.3 DATASETS FOR SOIL DEGRADATION ANALYSIS

In the following we present the datasets used in our study. Table 4.1 summarises the datasets used for the mapping procedure conducted with the U-Net which were also the basis for the training dataset produced with OBIA (Zweifel et al., 2019).

Table 4.1: Summary of raster datasets used in this study. All geodata sets © swisstopo. The aerial image of 2016 (⋆) was only used for testing the generalisation of the U-Net model.

Dataset	Derivative	Resolution	Recording Date	
Aerial Image		0.5 m	24 August	2000
(RGB bands)		0.5 m	9 September	2004
		0.25 m	20 July	2010
		0.25 m	1 August	2013
		0.25 m	20 July	2016⋆
Digital Terrain	Slope	2 m		
Model	Aspect	2 m		
(grey scale)	Curvature	2 m		

4.3.1 *Aerial Imagery*

The aerial images of SwissImage are high-resolution georeferenced orthophotos (product of Swisstopo (2010)). Five aerial images covering the Urseren valley were used in the time from 2000 to 2016. These images have a spatial resolution of 0.5 or 0.25 m (Table 4.1). Spectral information is available in the visible range (red, green, and blue spectral bands). All aerial images have slightly different properties (e.g. spatial resolution, colour distribution, and lighting conditions) but were always recorded during the growing season between late July and early September.

4.3.2 *Digital Terrain Model*

The digital terrain model (DTM) SwissALTI3D is the surface model of Switzerland without vegetation and development and has a spatial resolution of 2 m (product of Swisstopo (2014)). Based on the elevation information of the DTM we derived the slope, aspect, and curvature (plan and profile) using ArcGIS (Version 10.5). The DTM provides valuable information and offers context to the aerial images. Zweifel et al. (2019) have shown that for their study using OBIA, the DTM and its derivatives were essential for successful erosion mapping and classification.

4.3.3 *Training Data*

The data used to train the U-Net model consists of aerial imagery, DTM information, and training labels (see Section 4.4.3 for the training process). To train our U-Net model, a subsection (9 km^2) of the Urseren valley (26 km^2) was used with the corresponding OBIA-mapped features, see Figure 4.2. Four of the aerial images were used during training, leaving out the year 2016. By separating a subsection for training, we validated the spatial transferability of the model within the larger valley region. In addition, by omitting 2016, we tested the spatial and temporal transferability when applying the U-Net to a different image with properties not known during training.

4.3.3.1 *Training Labels*

The training labels come in the form of mapped erosion sites with attributed erosion classes from a previous study by Zweifel et al. (2019). This dataset was created with a semi-automatic method using an OBIA approach described in Section 4.4.1, which made use of the same aerial imagery and DTM information as used for the U-Net. Mapped erosion objects are available for the entire Urseren valley for all five aerial images (2000, 2004, 2010, 2013, and 2016). Based on random sample evaluation by experts, this dataset has an average overall accuracy score of 85.4% (Zweifel et al., 2019). The training labels consist of four different erosion classes which we introduced in Chapter 2 and which are illustrated in Figure 4.3: shallow landslides (areas with displaced topsoil layers and clear boundaries to the surrounding vegetation), livestock trails (elongated tracks caused by livestock trampling, mostly perpendicular to the slope), sheet erosion (patches with reduced vegetation cover), and management effects (large areas damaged by heavy machinery, over-fertilisation, or intense grazing in fenced-off areas).

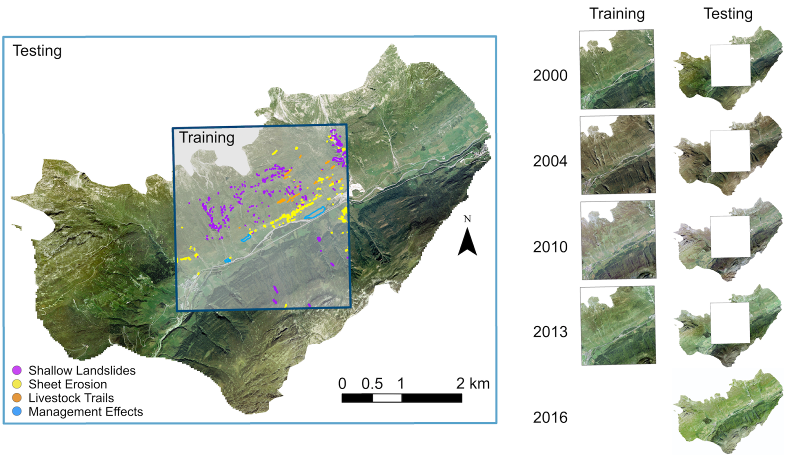


Figure 4.2: Training (9 km^2) and testing (17 km^2) areas are marked on the aerial image with examples of OBIA training labels for 2000 (map on the left). On the right-hand side is an overview of all available years and the sections used for training and testing. All training areas contain OBIA training labels (not shown) for the respective years (2000 – 2013). Training labels vary for each year due to the continuous evolution of soil erosion sites. The entire area of the image taken in 2016 was used only for testing.

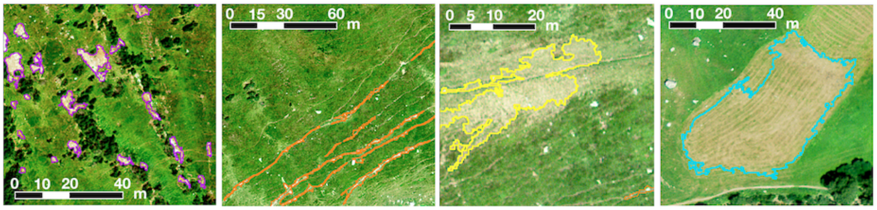


Figure 4.3: Examples for the labels used for training the U-Net model. From left to right: shallow landslides, livestock trails, sheet erosion, management effects.

4.4 METHODOLOGY

Our methodology consists of two major parts: the training process and the prediction process, with an overview depicted in Figure 4.4. To train the U-Net model we use OBIA labels together with the respective aerial image information (RGB) and DTM information for a dedicated training area (9 km^2). The U-Net assigns pixel-wise probability values and thus provides information about the likelihood of pixels belonging to a specific erosion class. Based on these probabilistic assignments, hard segmentations are produced by thresholding. The following sections will describe the methodology in further detail.

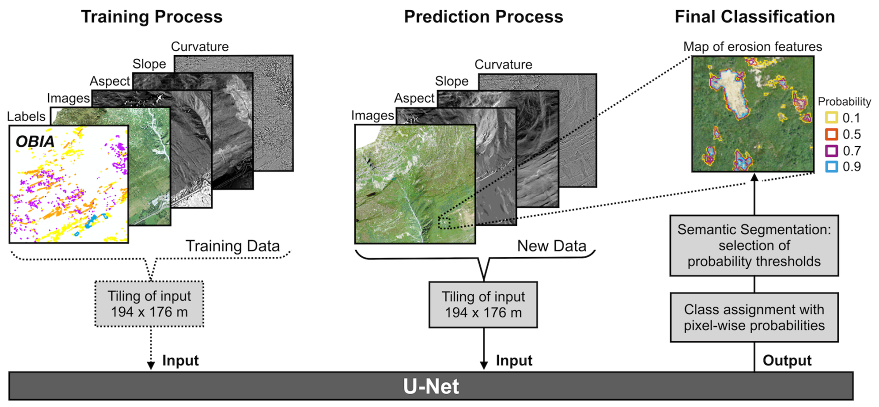


Figure 4.4: An overview of the developed workflow on the basis of the U-Net showing examples of input files for training and prediction purposes. The output shows one of four erosion classes, namely, shallow landslides, with four different probability thresholds.

4.4.1 *Object-Based Image Analysis*

Object-based image analysis (OBIA) combines a segmentation algorithm with classification techniques ranging from decision trees to various supervised machine learning algorithms which assign generated segments (or object primitives) to erosion classes. We used the software eCognition Developer (version 9.3.2) implementing a multi-resolution segmentation algorithm for grouping pixels with similar properties to object primitives. Input data consisted of aerial imagery (RGB), the excess green vegetation index, and information from the DTM and its derivatives (slope, aspect, and curvature). The object primitives contained information on their spatial, spectral, textural, and contextual properties based on all input data. Given these extracted feature sets, a random forest classifier was trained on manually selected samples in order to identify bare soil sites or sites with reduced vegetation cover. Subsequently, an additional decision tree was assigned specific erosion classes based on the typical appearances of objects previously identified containing bare soil or reduced vegetation cover. These erosion classes consist of shallow landslides, livestock trails, sheet erosion, and management effects. Note that the entire workflow needed to be performed on every input image to accommodate for varying image properties. Therefore, OBIA-labels for different input images can be considered to be obtained from independent models (i.e. differently calibrated settings). A detailed description of the workflow is presented in Zweifel et al. (2019).

4.4.2 *Neural Network Architecture*

In this study, we make use of the U-Net architecture (Ronneberger et al., 2015) illustrated in Figure 4.5. The U-Net is a fully-convolutional neural network which consists of a contracting part and an expansive part, i.e. an encoder-decoder architecture. We have introduced the U-Net in Section 3.6 and provide some more details relevant to our study in the following.

In the contracting part (upper part in Figure 4.5), a sequence of two convolutional layers with ReLU activations followed by max pooling layer processes the input.[1] With each max pooling application, the sizes of the resulting feature maps are halved, while the number of features is doubled for the subsequent convolutional layer. In the expansive part (bottom part), a sequence of transposed convolutional layers with ReLU activations followed by two convolutional layers and ReLU activations is applied to restore the original image size. Feature maps from the contracting part are appended to the feature maps obtained through the transposed convolutions to provide fine-detail features in the expansive part. Finally, a 1×1 convolutional layer followed by a pixel-wise softmax activation function provides the final segmentation output where each channel represents the segmentation map for the individual classes. The softmax function rescales the activations for each pixel to the $[0, 1]$ interval. More explicitly, for a pixel f in the output map F, the softmax yields a prediction $p_c(f)$ which can be interpreted as the probability of pixel f to belong to class $c \in \{1, ..., C\}$. The neural

1 Note that a common technique for standardising activations in CNNs and U-Nets is *batch normalisation*. However, in our experiments we did not notice an effect whether we used it or not. Thus, we do not employ batch normalisation.

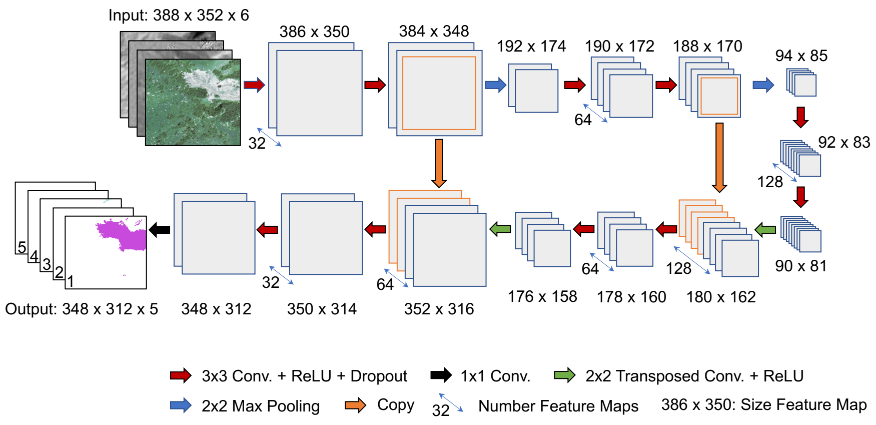


Figure 4.5: The employed U-Net architecture: In the first (upper) part, the input is contracted into a compressed representation (right). In the second (lower) part, the compressed representation is expanded into a segmentation map with pixel-wise class probabilities. The input consists of the input RGB image (three channels) and the DTM derivative maps for the aspect, curvature, and slope (one channel each). The resulting output provides a segmentation map for each considered class: Shallow landslides (indicated by 1 in the output), livestock trail (2), sheet erosion (3), management effects (4), and a class for non-assignable pixels (5).

network is trained with the cross-entropy loss which penalises incorrect class assignments with

$$-\frac{1}{N}\sum_{f\in F}\sum_{c\in C} y_c(f)\log(p_c(f)) \tag{4.1}$$

where $N = |F|$ is the number of pixels and $y_c(f)$ is the ground truth class assignment for pixel f, i.e. 1 if c is the correct class and 0 otherwise. For any pixel f in the input image, the softmax prediction $p(f) = (p_1(f), p_2(f), ..., p_C(f))$ provides the probabilities for the classes $c \in$ {shallow landslide, sheet erosion, livestock trail, management effect, non-assignable} – e.g.

$$p(f) = (\underbrace{0.55,\ 0.1,\ 0.2,\ 0.05,}_{\text{erosion class probabilities}}\ 0.1). \tag{4.2}$$

In addition to the four erosion classes, a class for non-assignable pixels is introduced which represents the class for all remaining (potentially ambiguous or vegetation covered and thus stable) objects. The U-Net provides pixel-wise class probabilities like in Equation (4.2) as the probabilistic output. In the following, for each erosion class we will refer to the full-probability result when only entries for the specific class of this output are considered without applying a threshold (e.g. the first entries for shallow landslides). For the final hard segmentation, we would like to obtain the dominant erosion class and apply different probability thresholds that control to which extent candidate segments are obtained. We only consider the erosion classes and identify the class with the largest probability for pixel f as the dominant erosion class. If the selected erosion class probability does not meet the threshold, the respective pixel is considered as a background pixel. For example, in Equation (4.2), $\operatorname{argmax}\{0.55,\ 0.1,\ 0.2,\ 0.05\}$ implies that shallow landslide is the dominant class and pixel f is predicted to be a shallow landslide pixel with a probability of 55%. At a threshold of 0.5, the class probability exceeds the threshold and pixel f is assigned to the shallow landslide class, while with a stricter threshold of 0.6 the pixel is considered to be a background pixel. With this kind of threshold segmentation, the final erosion class labels are obtained.

4.4.3 *Training Process*

In order to learn how to identify erosion sites, precise boundaries for the different erosion classes are required for training the U-Net. Inadequate training labels can deteriorate the spatio-temporal generalisation capability of the U-Net. In this study, we used high-quality training labels provided by the OBIA approach (see Section 4.3.3.1), and we considered the resulting erosion class areas as the ground truth segmentation in this investigation. To process the input images efficiently, we divided the aerial images into tiles of size 194×176 m which correspond to 388×352 pixels at 0.5 m resolution (2000, 2004) and 776×704 pixels at 0.25 m resolution (2010, 2013, 2016). The same is done for the maps of the DTM derivatives aspect, curvature, and slope.

Adjacent tiles overlap such that a 20 m (40 and 80 pixels, respectively) margin of one tile is contained in an adjacent tile. Figure 4.6 illustrates the resulting tiles for different years. The higher resolution tiles were down-sampled so that all input tiles are of size 388×352 pixels. No data augmentation was employed, as we expect object size and orientation (e.g. north/south exposure) to be relevant features. As described previously, the U-Net was trained from scratch with tiles extracted from the training area of the years 2000 to 2013, with a total of 1292 training samples. A U-Net of depth 3 with initially 32 (root) filters was used (see Figure 4.5), resulting in 467 525 network parameters. The network was trained for 300 epochs with a batch size of 20, using the Adam optimizer (Kingma and Ba, 2015b) with a learning rate of 0.001 and a dropout rate of 0.1. We used TensorFlow version 1.10 (Abadi et al., 2015) for our implementation which is based on the U-Net implementation by Akeret et al. (2017). The full source code of our analysis pipeline is available under the GNU General Public License v3.0.[2]

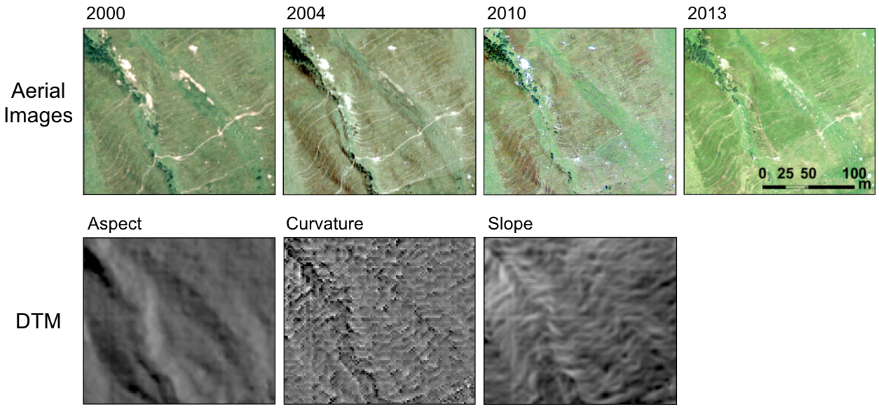


Figure 4.6: Example of input RGB images for training for the years 2000, 2004, 2010, and 2013 with a size of 194 × 176 m (corresponding to 388 × 352 pixels at 0.5 m resolution). The images show examples of eroded area on grassland slopes (livestock trails, shallow landslides). Below, the corresponding aspect, curvature, and slope maps are displayed (for all years the same DTM information is used). To obtain the samples, the aerial images of the respective years (see Figure 4.2) and the DTM derivatives were divided into smaller tiles.

4.4.4 *Details on the Evaluation*

For the evaluation, only sites with an area of at least 4 m^2 were considered, which we treated as the minimum reasonable object size. This is in line with the definition used in Zweifel et al. (2019). After choosing an appropriate probability threshold, the quality of the segmentation results was assessed with the precision score (producer's accuracy), recall score (user's accuracy),

2 https://github.com/bmda-unibas/ErosionSegmentation

and their harmonic mean, the F_1 score. We considered objects which overlap in both the OBIA and U-Net results as true positives and weight true positives, false positives, and false negatives by the areas of the respective segments. Ultimately, our goal was to evaluate the total degraded area on the held-out test area of the training years (2000, 2004, 2010, and 2013) and the test year 2016 in comparison to the OBIA ground truth results. The emphasis here was to study the temporal trend and relative increase in degraded area as obtained from the different methods. We performed a linear regression to provide the linear trend over the time period from 2000 to 2016.

4.5 RESULTS AND DISCUSSION FOR THE URSEREN VALLEY

The U-Net provides pixel-wise probabilities for each erosion class, which allows for assessing the certainty of predictions by studying the resulting heat maps (see Figure 4.7 for an example). In practice, this rich information is further post-processed by applying a threshold on the pixel-wise probabilities to form well-delineated segments. In the following, we present both results on the (full-probability) heat maps and results obtained with a selection of different probability thresholds. The latter enables a more direct comparison to the segmentation results obtained with OBIA. All results were obtained on the held-out test area (see Figure 4.2). Note that the data from 2016 was not used for training.

4.5.1 *Segmentation of Soil Erosion Sites*

The trained U-Net provides satisfying segmentation results which are demonstrated in Figure 4.7 for exemplary segments of shallow landslides and livestock trails. The heat maps illustrate the full-probability output of the U-Net and display the certainty in the class assignment (upper panel). By selecting different thresholds, hard class assignments can be achieved which lead to slightly different segment shapes depending on the threshold (lower panel). We selected thresholds of 0.2 and 0.8 to display the impacts of a wide range of probability thresholds on the delineation of segments. In general, choosing lower thresholds allows for the identification of a large number of potential erosion sites, while a higher threshold reduces the number of segments and also has an effect on the margins of these object, i.e. shrinks the segments to the most certain area. The probability threshold is a free parameter which can be chosen guided by application requirements or user preferences, or in our case to match baseline results (OBIA).

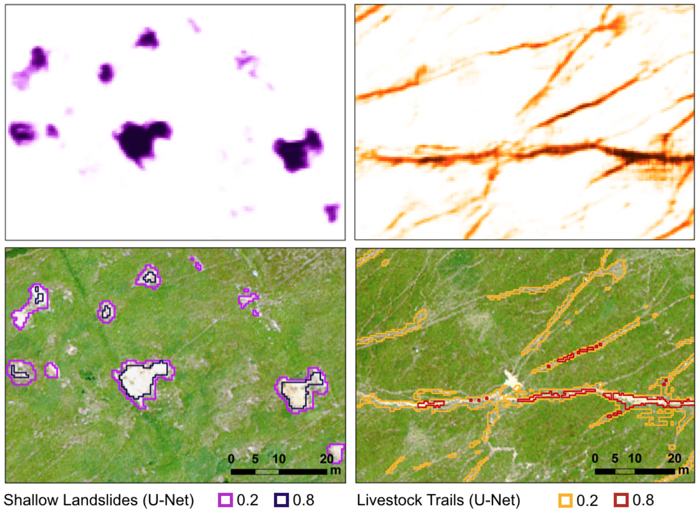


Figure 4.7: Visualisation of U-Net mapped shallow landslides (left) and livestock trails (right) for 2016. The lower panel shows segmentation results with different probability thresholds: the lighter colour indicates a lower probability threshold (0.2) and the darker colour indicates a higher probability threshold (0.8). Lower thresholds lead to larger and more numerous segments. For the same region (background omitted for better visualisation), the upper panel shows the full-probability heat map output of the U-Net: darker colours indicate higher probabilities.

In order to evaluate the accuracy of the proposed U-Net approach, we consider the OBIA results for 2016 as the ground truth baseline, which are independent of all other years, as OBIA was separately applied to the aerial image of 2016. For the comparison, we selected a threshold value of 0.3, as this led to the best agreement between U-Net and OBIA segments with respect to the total degraded area (see Section 4.5.3). OBIA relies on a dedicated, multi-resolution segmentation algorithm which provides clear objects to start with, which can then be classified. In contrast to OBIA, the U-Net approach does not have such a procedure and thus provides less control over segment shapes, as these are determined by pixel-wise thresholding. Consequently, there are cases in which both OBIA and U-Net identify areas as erosion sites but the boundaries of these objects might differ slightly. In that respect, our results for the U-Net show that erosion sites with clear, unambiguous boundaries such as shallow landslides (and some very clear cases of livestock trails) generally have better overlaps with the OBIA baseline and contiguous objects are better identified (see Figure 4.8 on the left). Boundaries of more diffuse erosion sites predicted by the U-Net show a slight mismatch with the OBIA baseline (see Figure 4.8 on the right). In these cases, the correct delineation of sites belonging to management effects or sheet erosion is in general a challenging task which is mirrored in the less accurate matching of the segmentation results from the different methods. Additionally, these erosion classes have similar appearances and are comprised of either bare soil or vegetation areas with strongly reduced vegetation cover which are prone to similar erosion processes (mainly erosion by water run-off), and they differ only in the origin of the damage. Management affected sites are mostly located near the foot of the slope, are mainly used for the production of hay, and can show signs of heavy machinery usage. Sheet erosion, on the other hand, can be found throughout the entire valley and can be caused not only by livestock trampling and grazing, but also climate-related factors, such as drought, precipitation, and snow-melt (Alewell et al., 2008; Meusburger and Alewell, 2014; Konz et al., 2010, 2012; Zweifel et al., 2019). Still, both methods are able to identify a great majority of overlapping objects. More quantitatively, we obtained scores for a threshold value of 0.3, as presented in Table 4.2.

Table 4.2: Scores for the U-Net with a threshold value of 0.3 for the test aerial image of 2016. U-Net results are compared to OBIA baseline results.

Scores	**U-Net**
Recall	84%
Precision	73%
F_1	78%

The precision score indicates that 73% of the predicted U-Net segments have corresponding OBIA segments, and about 27% of predicted U-Net segments do not directly correspond to any OBIA segments. On the other hand, the recall displays that 84% of the OBIA segments are maintained and the remaining 16% of OBIA segments are not identified by the U-Net. Both these findings suggest that the U-Net successfully identifies a majority of OBIA segments (recall score), but provides more segmented erosion sites than OBIA (false positives). Segments contributing to the 27% false positives can still be valid erosion sites which are not captured by OBIA, as

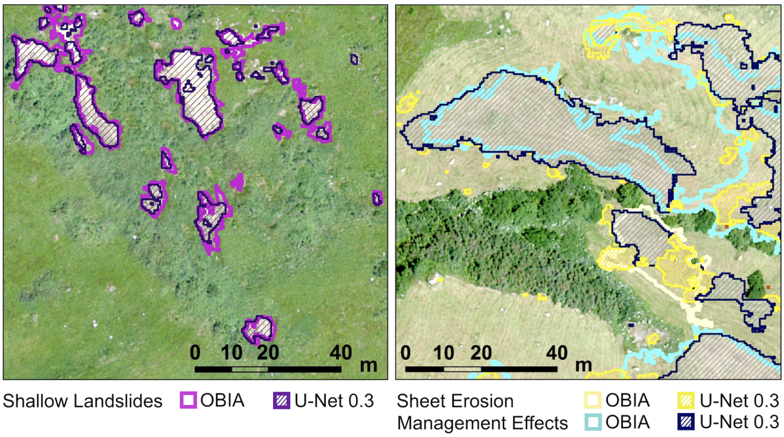


Figure 4.8: Comparison of segmentation results of OBIA and the U-Net (probability threshold of 0.3) for the aerial image of the year 2016. This aerial image was not used during training of the U-Net model and the depicted sections are located in the held-out test area. Lighter colours show OBIA results; darker colours (shaded) are results of the U-Net.

it is known that OBIA tends to give a conservative estimate of the degraded soil (Zweifel et al., 2019). Therefore, it is important to note that these scores mainly highlight the difference between U-Net segmentation with respect to the OBIA segmentation baseline, and it is possible that one method captures valid erosion sites which the other method misses (see example shown in Figure 4.9 on the right).

Most cases of false positive predictions can be related to objects which are similar in appearance to the erosion classes, and the reason for misclassification can be recognised in many cases upon manual inspection. False positives are typically patches with rocks located at higher elevations which are classified as shallow landslides (see Figure 4.9 on the left), or varied classification of sites affected by management and sheet erosion. Nonetheless, singular rocks on grassland areas are successfully left unclassified. These kinds of disagreements are inherent to the U-Net approach, which attempts to identify regularities in the training data and thereby includes objects which share some similarities. In clear cases, such as very small object sizes or predictions at certain altitudes where a particular class of erosion phenomena is not expected, a post-processing step can address these erroneous classifications. Another way of avoiding segmentation ambiguities is to employ pre-processing steps to identify sub-regions of interest for

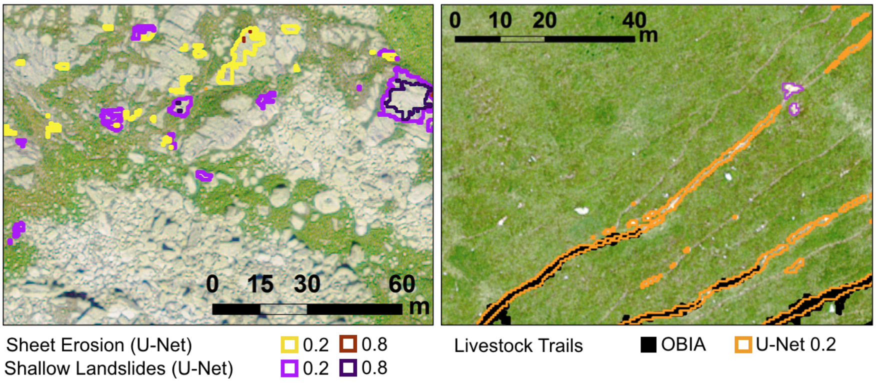

0 15 30 60 m
0 10 20 40 m
Sheet Erosion (U-Net) 0.2 0.8
Shallow Landslides (U-Net) 0.2 0.8
Livestock Trails OBIA U-Net 0.2

Figure 4.9: Examples of two different types of false positives: On the left-hand side, the U-Net identifies some rock surfaces as sheet erosion (yellow) and shallow landslides (purple). For both erosion classes, thresholds of 0.2 and 0.8 are shown. Lower threshold choices are linked to more of such false positives. Depicted on the right-hand side are livestock trails with OBIA and the U-Net (threshold of 0.2). Here, the U-Net is capable of identifying more livestock trails correctly compared to OBIA.

target objects which share some kind of regularity in their appearance, for instance, in the shape of the objects (Guirado et al., 2017). For the purpose of this study, however, no pre-processing steps were used in the U-Net procedure to ensure objective comparison with OBIA.

4.5.2 *Threshold Selection*

In similar studies, the matter of threshold selection is usually not addressed or a fixed threshold value is used. This can be suitable for studies with binary output classes (e.g. (Baumhoer et al., 2019; Zhang et al., 2018c)), but can also be problematic for gradual transitions of classified objects, as discussed by Kattenborn et al. (2019). Other studies employ deep learning approaches for classification of the object primitives in the OBIA framework where object boundaries are already well-defined (Fu et al., 2019; Zhang et al., 2018a; Lu et al., 2020). In our setting, threshold selection can be used to adjust segmentation results in relation to pre-existing knowledge (i.e. segmentation results of other methods such as OBIA), which led to the best fit with a threshold selection of 0.3 for this study (with respect to the total degraded area; see Section 4.5.3). Additionally, varying thresholds may be applied to make necessary adjustments for different classes with varying appearances. As a standard comparison, a held-out dataset of the ground truth segmentation required for training can be used to determine appropriate probability thresholds if necessary. In the absence of appropriate pre-existing knowledge or in cases where visual assessment is not possible, it is advisable to use a range of probability thresholds which capture a variety of segment estimations and assess uncertainty ranges of the estimates.

4.5.3 *Trend Analysis of Soil Erosion Sites*

In order to study the temporal trend in the extent of soil degradation, we applied the U-Net to the series of five aerial images of the Urseren valley between 2000 and 2016 (see Section 4.3.3.1). We compare the full-probability U-Net results and the results for the different thresholds to the baseline results of the OBIA approach in Figure 4.10. In the first case, the heat map results are added up to form an estimate of degraded area per erosion class. The resulting outcomes of the full-probability U-Net output match the OBIA results closely with respect to the total degraded area. Due to their methodological differences, slight deviations in the segmentation results and the resulting (total) degraded area were expected. The same holds true for the U-Net results with a threshold of 0.3. This threshold was identified to exhibit the most suitable agreement with OBIA segmentation results with respect to the total degraded area. It can be observed that for test year 2016, the OBIA and U-Net threshold 0.3 results agree very well (in the shaded area in right plot of Figure 4.10). As expected, the U-Net results display an increase in degraded area for decreasing thresholds. Nevertheless, in all considered U-Net results, the same temporal trends of decrease and increase from one year to another are observed, as in the OBIA baseline. This observation is also supported by the linear regression results, which in all cases provide similar linear temporal trends. In order to quantify the relative increase in degraded area, we consider the values for 2000 and 2016 obtained from the linear regression line. Again, the threshold dependency with respect to the total degraded area is observed (top panel in Figure 4.11). However, for the relative increase in degraded area (quotient of values for 2016 and 2000), the results become mostly independent of the selected threshold (bottom panel in Figure 4.11). To assess the statistical uncertainty of the linear regression fit and thus the relative increase, one standard deviation each of the fitted parameters (slope

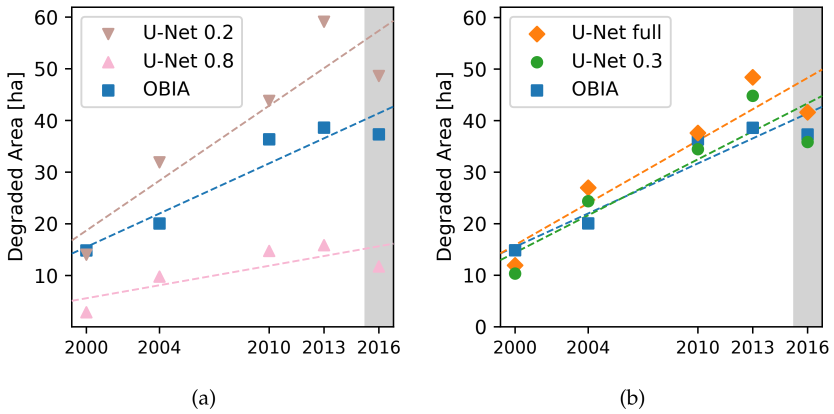


Figure 4.10: Linear trend of the total degraded area in the held-out test region (see Figure 4.2). (a) Results for a range of different threshold values in the U-Net approach as compared to the OBIA baseline. (b) Results for the suitable threshold value 0.3 and the full-probability results are given. Qualitatively, a similar increase or decrease of degraded soil in the individual years is retained in all models. The linear interpolation provides a similar temporal trend of increase in degraded soil in all cases. In particular, the full-probability and threshold 0.3 results of the U-Net approach show good agreement with the OBIA baseline. The linear trends with lower and higher thresholds surround the OBIA result. The years 2000 to 2013 provide a result on the spatial generalisation of the U-Net (years used for training), while the result for 2016 (shaded column) in addition provides a temporal generalisation result (aerial image of 2016 was not used for training). Note that the OBIA approach needs to be trained on all aerial images.

and intercept) is considered to obtain the two most extreme linear trends which are possible within the uncertainty of the fitted parameters. This means the steepest and flattest linear trends with respect to one standard deviation in the parameters are identified, which leads to the error bars for the total degraded area as depicted in Figure 4.11. As the relative increase considers the ratio of these quantities, the error bars are relatively larger for the relative increase of degraded area. In particular, for a threshold of 0.8, the statistical uncertainty increases due to the comparably small degraded area detected. The obtained U-Net results show similar relative increases of degraded area which fall within the uncertainty range of each other depicting the statistical uncertainty in the linear regression fit (one standard deviation). The U-Net results are in good agreement

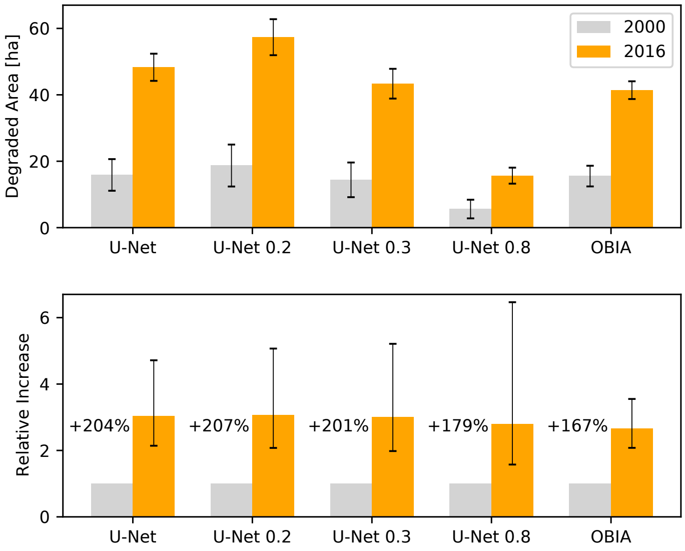


Figure 4.11: Comparison of total degraded area in years 2000 and 2016 for the baseline (OBIA) and the U-Net approach with different thresholds. The total degraded area was obtained from the interpolation results of each year (top panel). In all approaches, an increase of degraded area in the Urseren valley is observed with threshold-specific differences in the total extent. However, the relative increase in degraded area (bottom panel) shows that assessing the trend of soil degradation can be done independently of the threshold, as all results fall within the statistical uncertainty of the linear regression fit. Note that the statistical uncertainty for U-Net 0.8 increases due to the comparably small total degraded area detected. The error bars depict the statistical uncertainty of one standard deviation.

compared to the baseline method, with an increase of 167% in the test region. This in turn is in line with the increase of 156% ± 18% reported in Zweifel et al. (2019) for the full Urseren valley, where ±18% depicts the estimated propagated error based on expert accuracy assessment (and not the statistical uncertainty in the linear regression fit). Importantly, it has been established that OBIA tends to underestimate the extent of degraded soil (Zweifel et al., 2019). Therefore, the steeper relative increase obtained by the U-Net results is plausible and potentially reflects the increase of degraded area more accurately. Furthermore, the fact that the relative increases for the different probability thresholds coincide with each other within the statistical uncertainty of one standard deviation of the linear regression fit is further evidence for the applicability and robustness of the U-Net approach. Assessing the relative development of aggregated measures, such as the total area of degraded soil, is therefore less sensitive to the choice of threshold. The results on the linear trend (Figure 4.10) and the relative increase of total degraded area (Figure 4.11) highlight that the probabilistic output of the U-Net aligns with the OBIA results very well. Thus, it is not necessarily required to study these quantities by choosing a threshold, i.e. hard segmentation. In our investigation we assess predictions in the held-out test region (see Figure 4.2) for two validation cases: (i) testing the erosion site prediction of the test region for years for which conditions (colour, shading, vegetation, etc.) were available during training (2000 – 2013) and (ii) testing the predictions for a new test year for which conditions were unknown during training (2016). In the first case, our results provide evidence that the trained U-Net transfers well to adjacent regions with similar conditions, as observed during training, which demonstrates the *spatial generalisation* capability of the U-Net approach. Furthermore, the latter validation case gives evidence of suitable erosion site segmentation with the U-Net approach in completely new aerial images with conditions not encountered during training, which in addition highlights the *temporal generalisation* capability of the approach. For the individual erosion classes,

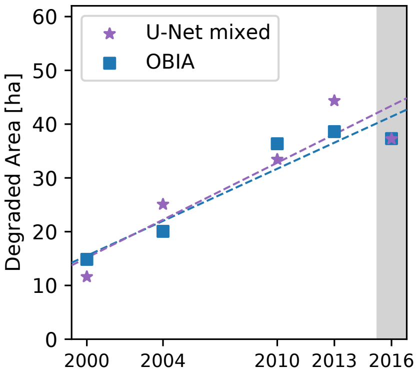
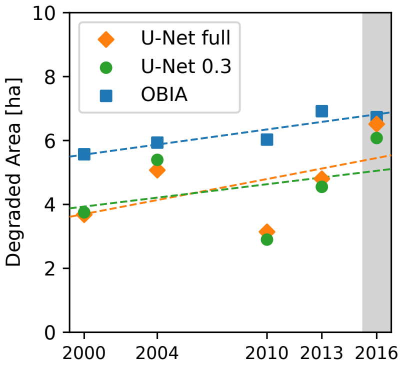


Figure 4.12: Total degraded area prediction of the U-Net with individually selected thresholds best suited for every erosion class on the held-out test region. The thresholds were selected according to a detailed threshold analysis (not shown) to be: 0.2 for shallow landslides, 0.2 for livestock trails, 0.3 for sheet erosion, 0.5 for management effects. Although deviations in the total degraded area persist, the linear trends of the two methods almost coincide.

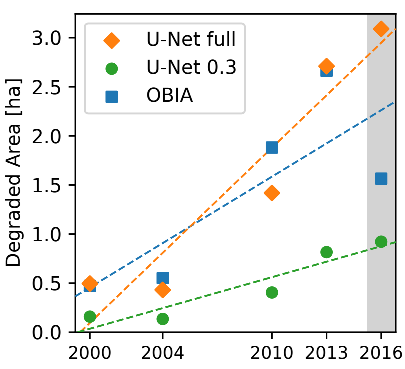
we examine the results for the full U-Net model output and for a threshold of 0.3 (see Figure 4.13). Especially for sheet erosion and management effects, which contribute to a great amount of the total degraded area, the choice of 0.3 as a threshold for the hard segmentation is appropriate. In the case of livestock trails, the full-probability U-Net results capture the behaviour in the baseline more appropriately. The individual results highlight that an erosion-class-specific choice of the probability threshold can be reasonable in applications such as ours. We provide a result on such a mixture of thresholds for the linear trend for the years 2000 to 2016 in Figure 4.12. The linear trend for the years 2000 to 2016 exhibits good agreement with the OBIA baseline (similar to Figure 4.10 on the right). Therefore, although the temporal development of aggregated measures is less dependent on the threshold, choosing different probability thresholds enables flexibility in the number of identified segments and segment boundaries in the U-Net approach. This is especially the case when examining the temporal development with regard to the degraded area per individual erosion class (Figure 4.13).

4.5.4 *Deep Learning and OBIA*

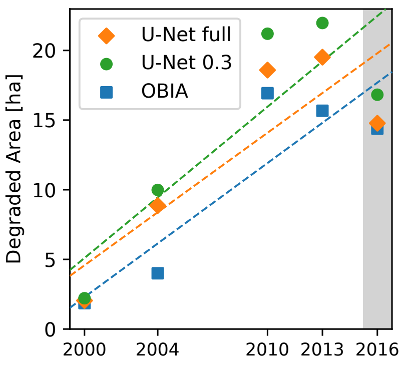
Deep learning methods for similar applications are predominantly trained with manual labels, and often the objects of interest are precisely defined, such as roads, buildings, or damaged trees in forests (Zhang et al., 2018c; Mboga et al., 2019; Hamdi et al., 2019). In our application, the objects are less clearly defined, and some of the segment boundaries concerning both the mapped and omitted areas might be more disputable. The boundaries of objects are often ambiguous due to smooth transitions, especially for erosion sites with reduced vegetation cover. Imprecise delineation of the objects of interest negatively impacts the generalisation capability and applicability of deep learning techniques, and can potentially be a limiting factor for this kind of approach. In particular, it can have a detrimental effect on the accuracy of the U-Net approach if the ground truth misses a great number of relevant objects. Therefore, we do not rely on manual labels of the objects of interest, which might suffer from subjective assessments, require labour-intensive work, and usually are unable to achieve pixel-level precision. Instead, we showcase that any kind of segmentation technique, such as OBIA in our study, can be used as a basis to provide training data to successfully employ a convolutional neural network for segmentation of natural features, such as the erosion sites in our application.



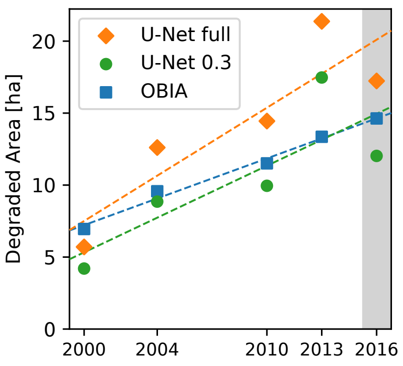
(a) Shallow Landslides



(b) Livestock Trails



(c) Management Effects



(d) Sheet Erosion

Figure 4.13: Mapped degraded area in the test region by erosion class for both the OBIA and U-Net methods (full-probability results and threshold value 0.3). Comparing the two methods, class-specific differences for the yearly degraded area and linear trends can be observed. Moreover, by selecting appropriate thresholds for each erosion class, similar linear trends in both methods can be attained (see Figure 4.12). The years 2000 to 2013 provide a result on the spatial generalisation of the U-Net (years used for training), while the result for 2016 (shaded column) in addition provides a temporal generalisation result (aerial image of 2016 was not used for training).

Similar studies have compared OBIA to deep learning approaches for the detection of landslides on remotely sensed data with the goal of enabling large-scale analysis. In Prakash et al. (2020) the comparison was done on the basis of landslide inventories. A study of different machine learning and deep learning methods was conducted by Ghorbanzadeh et al. (2019), who used field observations with manual corrections as the ground truth segments. These studies show that deep learning approaches improve segment detection by comparison of the segmentation performances of the different methods. In our study, we leverage the fact that OBIA is a well-suited approach for segmentation tasks on small scales, and thus derive our baseline trends and the ground truth segments from it. Other work like the detection of shrubs on high resolution satellite imagery by Guirado et al. (2017) similarly shows that CNN approaches can outperform OBIA in certain cases. That study relied on manually delineated ground truth segments and used dedicated pre-processing steps to identify regions of interest to perform classification of candidate patches. Combining OBIA and CNN approaches was also studied with regard to using CNNs in the classification step of the OBIA framework (Fu et al., 2019) or using features learned by the CNNs to improve inputs to the OBIA workflow (Pan et al., 2019). In our study, OBIA provides the necessary high-quality ground truth segmentation, but our workflow is not bound to OBIA, and any other reliable approach can be used for this too.

The presented results of this study substantiate that the U-Net approach can perform on a par with OBIA. Moreover, the transferability to new data, the insensitivity of trends in aggregated measures to threshold selection, and the flexibility of fitting the U-Net results to existing knowledge or competing segmentation methods – apart from manual inspection of segmentation results – render the proposed approach advantageous for a great variety of applications. Furthermore, large-scale analysis is facilitated by improved running times. For training and prediction, an Nvidia GeForce Titan X Pascal GPU was used. In our study, training required approximately 6.5 h, while the prediction for the full Urseren valley took 12 min. This is a significant improvement over the semi-automatic OBIA approach, which takes up to a few days to achieve satisfying results for the Urseren valley. For large-scale studies (e.g. alpine-wide analysis) the U-Net-based process can efficiently be parallelised using several GPUs, resulting in even faster prediction times.

4.6 MULTI-VALLEY ANALYSIS

Studying soil degradation in the Swiss Alps requires segmentation of a great variety of diverse sites with different conditions and driving factors for the occurrence of erosion phenomena. Based on the U-Net solely trained on the Urseren valley, such an investigation of causal factors for shallow landslides in different grassland regions of Switzerland was performed in Zweifel et al. (2021). Ideally, a tool for mapping on an alpine-wide scale reflects these diverse settings while training the method. For this purpose, we extend the previous investigation by considering nine representative grassland regions in a *multi-valley* analysis and compare to the *single-valley* analysis of the Urseren valley in the previous sections. The location of the considered sites is illustrated in Figure 4.14, which include seven alpine regions, one foothill region (Hornbach valley) and one region in the Swiss Jura mountains (Baulmes) adopted from Zweifel et al. (2021). The selected sites vary in area, elevation, and orientation of the valley, with an overview given in Table 4.3. As in the single-valley analysis, the Urseren valley is included as one of the regions.

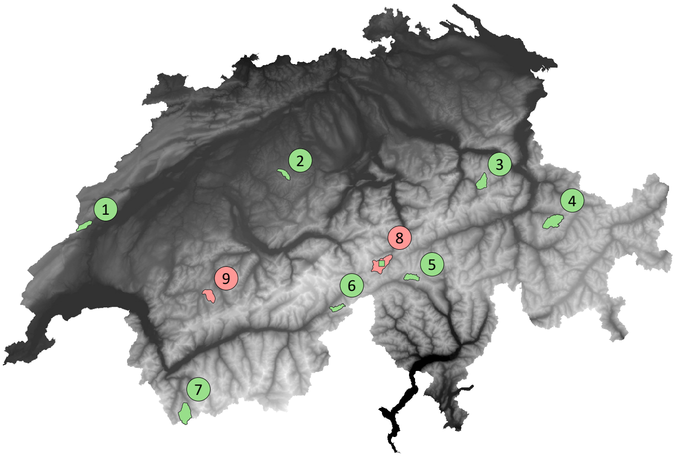


Figure 4.14: Overview of valleys considered in the multi-valley analysis: (1) Baulmes, (2) Hornbach, (3) Chrauch, (4) Arosa, (5) Piora, (6) Rappe, (7) Entremont, (8) Urseren, (9) Turbach. Green sites were used during training, red only for evaluation. Higher elevations are indicated by lighter colours.

Table 4.3: Summary of sites in the multi-valley analysis. All aerial images have a pixel resolution of 0.25 m, except for aerial images of the Urseren valley for 2000 and 2004 with 0.5 m pixel resolution.

	Study Site	**Area (km^2)**	**Elevation (m a.s.l.)**	**Orientation of Valley**	**Recording Year**
1	Baulmes	21	615 – 1512	NE/SW	2014
2	Hornbach	17	800 – 1256	NW/SE	2015
3	Chrauch	32	1421 – 2432	N/S	2014
4	Arosa	50	1613 – 2535	NNE/SSW	2014
5	Piora	21	1848 – 2554	E/W	2015
6	Rappe	16	1427 – 2533	NE/SW	2015
7	Entremont	50	1808 – 2823	N/S	2013
8	Urseren	26	1514 – 2840	NE/SW	2000 – 2016
9	Turbach	28	1208 – 2367	NNW/SSE	2013

4.6.1 *Adjustments in Methodology*

In the following, we make some adjustments in the training process and the dataset. In the single-valley analysis, it became apparent that low (relative) coverage of particular erosion classes in the OBIA ground truth segmentation led to low prediction probabilities for these classes. Livestock trail segments, for instance, tend to be under-represented and more affected sites can be identified (see Section 4.5.3). In order to reflect this in the optimisation process, we adjust the categorical cross-entropy loss in Equation (4.1) with class weights, i.e.

$$-\frac{1}{N}\sum_{f\in F}\sum_{c\in C} w_c y_c(f)\log(p_c(f)). \tag{4.3}$$

The class weights are chosen according to the frequency of labelled pixels of the respective erosion class in relation to all labelled pixels, attributing lower weights to more prominently covered classes. We identify weights $w_c \in \{0.2998, 0.4485, 0.1211, 0.1206, 0.01\}$ for classes $c \in$ {shallow landslide, livestock trail, management effect, sheet erosion, non-assignable}, i.e. we include an additional weight of 0.01 to the dominant class of non-assignable (background) pixels.

As a change in the architecture (see Figure 4.5), we use *same padding* in convolution layers, i.e. feature maps are padded with zeros ensuring that the feature map size is conserved after each convolution. This leads to output predictions being of the same size as the inputs and no overlapping tiling of patches is required (as described in Section 4.4.3). We find that this change from valid padding (i.e. no padding) to same padding does not have an influence on the overall segmentation results.

For convenience, input aerial images (as well as the DTM derivatives) are divided in 300×300 pixels input patches. For training, only patches with at least one annotated erosion site are used. In other words, patches in which all pixels belong to the non-assignable (background) class are excluded from the training set. No down-sampling is applied in the multi-valley analysis, i.e. all images preserve their original pixel resolution, which in most cases is 0.25 m. Only for the aerial images of the Urseren valley of 2000 and 2004 a pixel resolution of 0.5 m is used. The images for the DTM derivatives with 2 m pixel resolution are up-sampled to match patch resolution. We train on eight of the nine considered valleys (green sites in Figure 4.14), where we again divide the Urseren valley in a train and test area (see Figure 4.2). We exclude both the Urseren valley aerial image of 2000 and 2016 from the training. Additionally, we choose the Turbach valley as a validation region due to its spatial separation and almost orthogonal valley orientation (NNW/SSE) to the Urseren valley (NE/SW). These choices lead to a dataset of 9263 training images and 1899 validation images (Turbach valley only) which correspond to areas of 55 km^2 and 11 km^2, respectively. Furthermore, we test on the excluded test area of the Urseren valley (17 km^2) and have two test cases with different pixel resolution (0.5 m for 2000, 0.25 m for 2016). As before, no data augmentation is applied and most hyperparameters are chosen as in the single-valley analysis (see Section 4.4.3). We train the U-Net for 300 epochs with an initial learning rate of 0.001 but divide by a factor of ten every 200 epochs and reduce the batch size to 8 samples per iteration.

4.6.2 *Results and Discussion of Multi-Valley Analysis*

In the following, we evaluate the temporal trends of soil degradation in the Urseren valley similar to the single-valley analysis. For this, the U-Net was trained in ten independent reruns with the same hyperparameters and the mean degraded area over all ten runs is considered.[3] Furthermore, we take a closer look at the segmentation results for the Urseren valley (2016) and Turbach valley (2013), i.e. for test aerial images not considered during training. For visualisation, we choose the segmentation results of the training run which achieved the highest test accuracy.

4.6.2.1 *Temporal Trends in the Urseren Valley*

We focus on the same setting as in the single-valley analysis and study the five aerial images of the Urseren valley between 2000 and 2016 on the test area (see Figure 4.2). As before, we consider the full-probability prediction of the U-Net in both the single-valley and multi-valley setting, as well as the threshold of 0.3 which was identified as an appropriate threshold in the previous analysis. Figure 4.15a provides a comparison of the multi-valley results and Figure 4.15b a comparison of the single-valley and multi-valley setting for the 0.3 threshold to the OBIA baseline.

Generally, a larger total extent of degraded area is predicted in the multi-valley setting. The predictions in the single-valley setting follow much more closely the OBIA baseline. In all cases, increasing linear trends are obtained, with the results for the years 2010 and 2013 deviating more markedly from the linear regression line. On the basis of the linear interpolation, we obtain relative increases of total degraded area of 132% (0.3 threshold) and 130% (full-probability) from 2000 to 2016 in the multi-valley setting on average. These relative increases are close to the OBIA baseline result of 167% and fall within the uncertainty of the OBIA and single-valley results (see Figure 4.11). Taking a closer look into the specific erosion classes provides a similar picture illustrated in Figure 4.16. Generally, for all erosion classes larger predictions of degraded area are achieved in the multi-valley setting. This is particularly the case for sheet erosion. Considering again the linear interpolation for the relative increase of degraded soil per erosion class from 2000 to 2016, we obtain relative increases of $\{26\%, 114\%, 893\%, 125\%\}$ (0.3 threshold), and $\{40\%, 158\%, 716\%, 105\%\}$ (full-probability) in the multi-valley setting (on average) compared to the OBIA baseline results of $\{23\%, 396\%, 671\%, 103\%\}$ for shallow landslides, livestock trails, management effects, and sheet erosion.

3 The multi-valley results in Figures 4.15 and 4.16 show the sample means with error bars indicating the sample deviation. The linear trend is obtained with respect to the mean results.

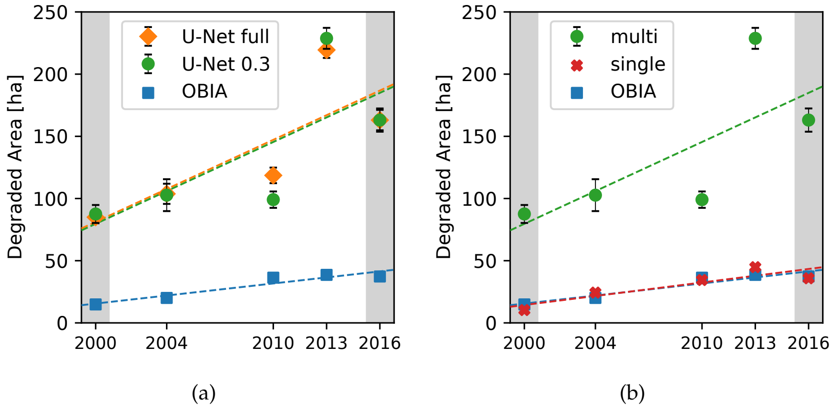
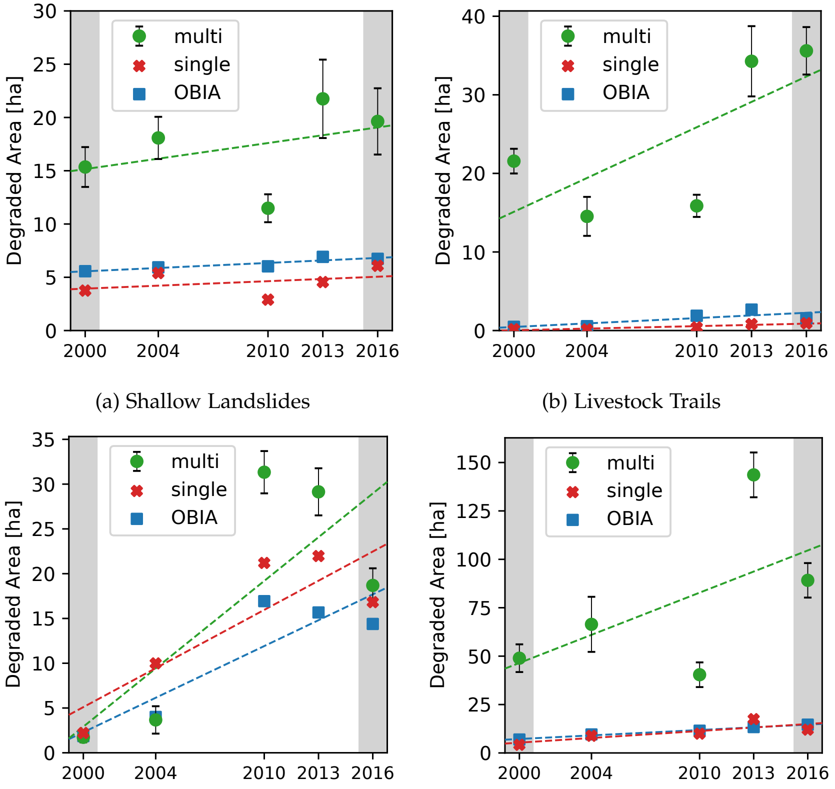


Figure 4.15: Linear trend of the total degraded area in the held-out test region of the Urseren valley (see Figure 4.2). (a) Results for the U-Net trained in the multi-valley setting (mean of ten independent runs) for full-probability and 0.3 threshold outputs compared to the OBIA baseline. (b) Comparison of U-Net predictions for 0.3 threshold in the (original) single-valley (same as in Figure 4.10b) and (new) multi-valley training setting. The multi-valley setting leads to generally larger degraded area predictions. Considering the linear interpolation in the multi-valley setting, we obtain a relative increase of 132% on average (0.3 threshold) close to the OBIA baseline result of 167% (see Table 4.11). Aerial images of years 2000 and 2016 were not considered during training (shaded columns).

4.6.2.2 *Segmentation Results in Multi-Valley Setting*

These results suggest that training in the multi-valley setting leads to over-segmentation compared to the OBIA baseline (and also the single-valley setting), with more sites being segmented as erosion phenomena. In particular, a large increase in sites segmented as sheet erosion is observed, which contributes significantly to the predicted total degraded area. Additionally, a larger extent of livestock trails is identified. However, the linear relative increases of the multi-valley setting show in most cases a good correspondence to the OBIA baseline. But the deviation of the trend line is more strongly pronounced, in particular for the results of the years 2010 and 2013. This might be related to the fact that most training patches come from aerial images recorded in 2013 to 2015 resulting in a more strongly pronounced imbalance towards this time window than in the single-valley analysis. On the other hand, different pixel resolutions (0.5 m for 2000 and 2004, 0.25 m



(a) Shallow Landslides

(b) Livestock Trails

(c) Management Effects

(d) Sheet Erosion

Figure 4.16: Mapped degraded area in the Urseren valley test region by erosion class for both the (original) single-valley and (new) multi-valley training setting (mean of ten independent runs), as well as the OBIA baseline. U-Net results for the 0.3 threshold are shown. Generally, the multi-valley setting leads to larger predictions for degraded area, in particular for sheet erosion. Considering the linear interpolation in the multi-valley setting, we obtain relative increases of $\{26\%, 114\%, 893\%, 125\%\}$ (on average) compared to the OBIA baseline results of $\{23\%, 397\%, 671\%, 103\%\}$. Aerial images of years 2000 and 2016 were not considered during training (shaded columns).

for all other) do not seem to have a large impact on segmentation results, with the relation of increasing or decreasing (total) degraded areas being similar between the results in the multi-valley setting as compared to the relations in single-valley setting.

Studying the segmentation results more qualitatively documents the observed over-segmentation well. Exemplary sites for all erosion classes are illustrated in Figures 4.17 (livestock trails), 4.18 (sheet erosion), 4.19 (shallow landslides), and 4.20 (management effects) for the Turbach and Urseren valley. On the one hand, as already noted in the single-valley analysis, additional segments are often plausible and extend the OBIA segments to potentially missed erosion sites. Therefore, considering these examples as false positives might not be justified or at least disputable. This is particularly the case for livestock trails, which are more and often better covered in the U-Net predictions of the multi-valley setting (see examples in Figure 4.17). This can be related to the fact, that occurrences of the geometrically more regular shaped livestock trails in different valleys with different orientations allow the U-Net to generalise better for this erosion class. On the other hand, however, we observe a clear over-segmentation of sheet erosion sites. This highlights the character of the erosion class which comprises sites of disturbed vegetation with generally diffuse boundaries which make clear assignments to this class more challenging. Similarly to the single-valley analysis, areas with rocks at higher elevations and sites of disturbed soil which cannot be clearly assigned to the other classes are typical examples of false positive segments of sheet erosion (see examples in Figure 4.18). Some of these false positives can be addressed by a post-processing which eliminates more rocky areas at higher elevations which are no longer grassland areas. Furthermore, a separate and more careful consideration of the sheet erosion class can be conducted in a down-stream application.

More quantitatively, for the test aerial image of the Urseren valley in 2016, 94% $\pm$ 1% of the segmented pixels in the single-valley setting are predicted as erosion site pixels in the multi-valley setting, too. Furthermore, we compare the U-Net segmentation results for the complete Urseren valley of 2016 as well as the Turbach valley of 2013 to the OBIA baseline. Table 4.4 summarises the scores. It should be noted that we adjusted the computation of these scores in the multi-valley setting. We consider pixels to be true positives only where U-Net and OBIA segments overlap perfectly. All pixels missing in the U-Net segmentation are consider false negatives and all additionally U-Net-segmented pixels are viewed as false positives. This is a

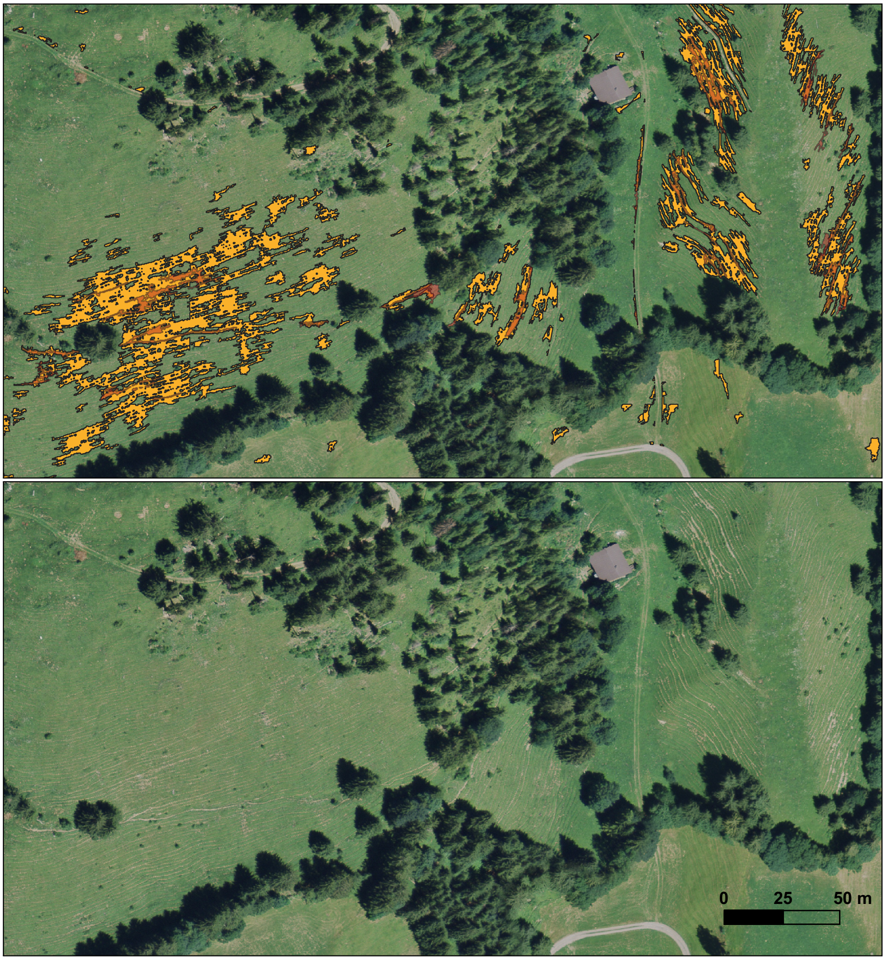


Figure 4.17: Livestock trails are often more appropriately covered in the U-Net segmentation results (light orange) than in the OBIA baseline (transparent red). Segmentation results (top) are shown for an exemplary site (bottom) in the Turbach valley (2013) which was not used for training. Note that the OBIA segments overlay U-Net segments which generally overlap well.

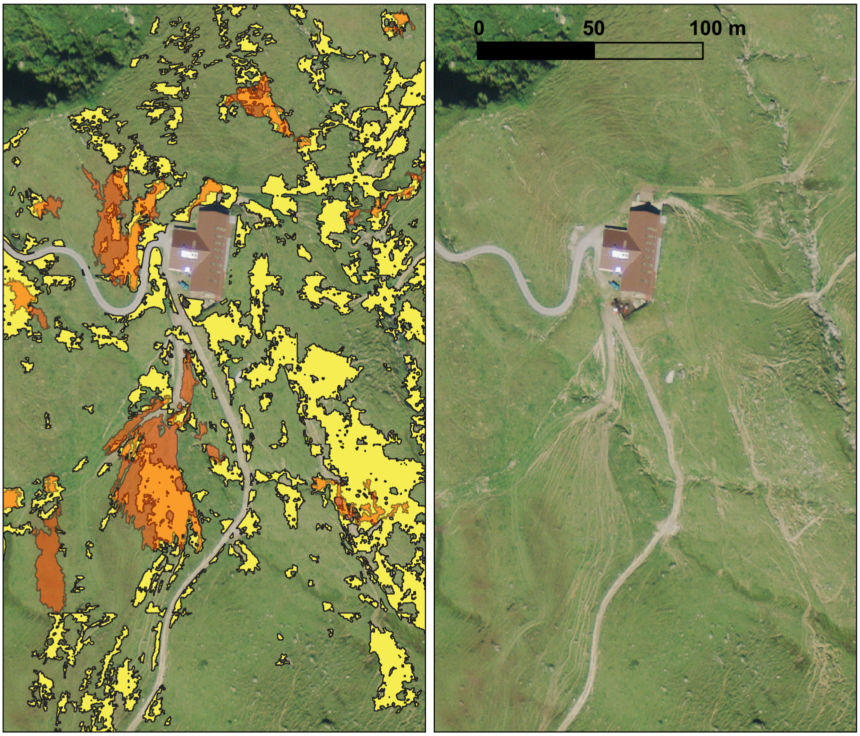


Figure 4.18: More sites are identified as sheet erosion in the U-Net segmentation (yellow) than in the OBIA baseline (transparent orange). Segmentation results (left) are shown for an exemplary site (right) in the Turbach valley (2013) which was not used for training. Note that OBIA segments overlay U-Net segments and differences are more noticeable.

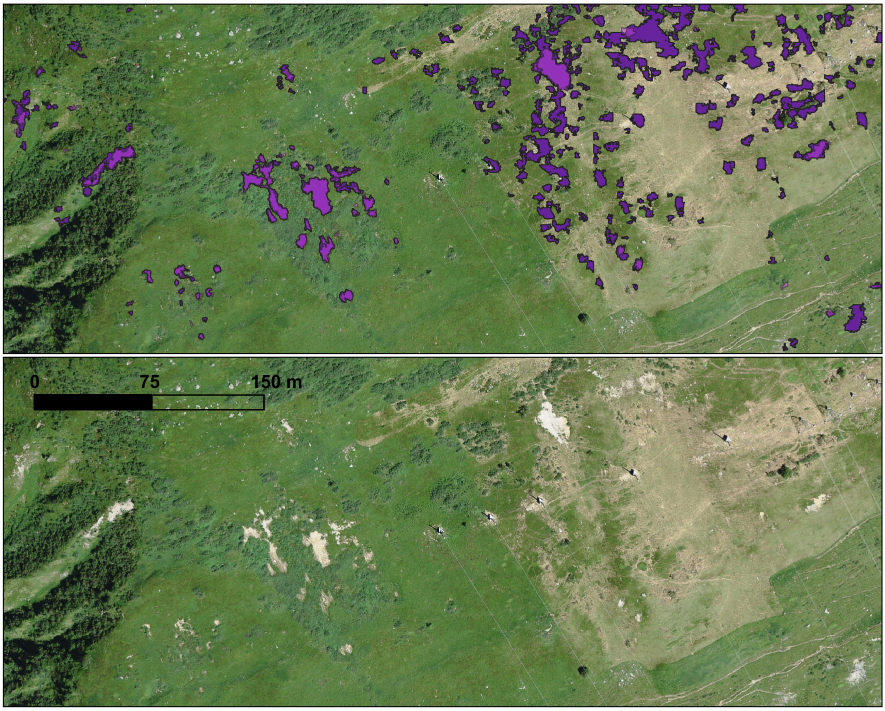


Figure 4.19: Shallow landslides segments agree well in both the OBIA baseline (transparent pink) and the U-Net segmentation (dark purple), which generally identifies more sites. Segmentation results (top) are shown for an exemplary site (bottom) in the test area of the Urseren valley for the test year 2016 which was not used for training. Note that the OBIA segments overlay U-Net segments and generally overlap well. Differences typically consist in larger U-Net segments which contain the OBIA segments.

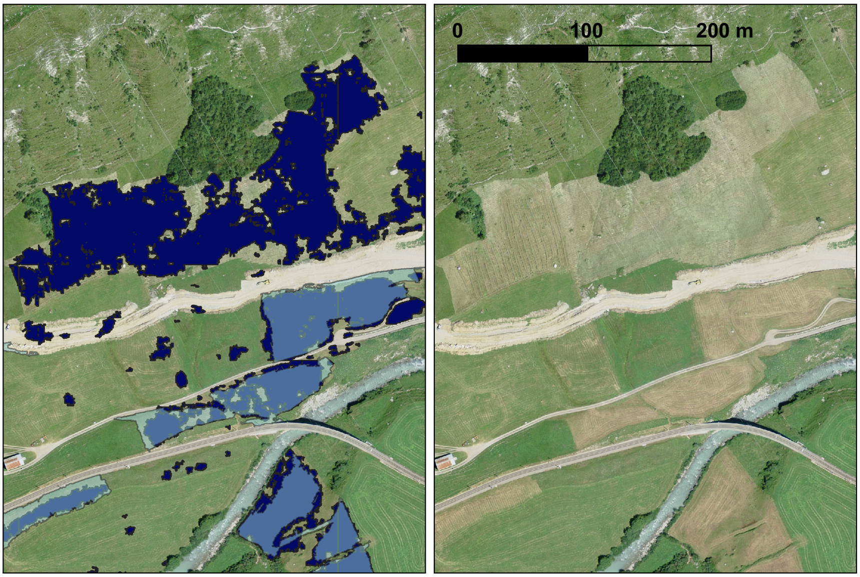


Figure 4.20: Grassland areas affected by management display a good agreement of the U-Net segmentation (dark blue) with the OBIA baseline (transparent cyan). Segmentation results (left) are shown for an exemplary site (right) in the test area of the Urseren valley for the test year 2016 which was not used for training. Note that the OBIA segments overlay U-Net segments and identify similar sites affected by management, but more sites being recognised by the U-Net prediction.

stricter definition of true positive than in the single-valley analysis, where the average area between overlapping segments was considered as true positive to account for different shapes of otherwise matching segments. In the single-valley definition, the recall score will be close to 100% due to relatively over-segmentation in the multi-valley U-Net results. This is also the reason why low values for the precision in both valleys are attained.

Table 4.4: Scores in the multi-valley setting with a threshold value of 0.3 for the test aerial images of Urseren 2016 and Turbach 2013. For both valleys, U-Net predictions are compared to the OBIA baseline segmentation. The scores were obtained for all ten individual runs and sample mean and sample standard deviation are reported.

Scores	Urseren	Turbach
Recall	$84\% \pm 1\%$	$68\% \pm 4\%$
Precision	$20\% \pm 1\%$	$22\% \pm 2\%$
F_1	$33\% \pm 1\%$	$34\% \pm 2\%$

In the multi-valley setting, training with an Nvidia GeForce Titan X Pascal GPU (as in the single-valley setting) took about 40 hours due to the larger training set. As before, obtaining the prediction matrix takes tens of minutes, while storing the results in commonly used shape files can take about an hour for the aerial images of the Turbach or Urseren valley at a pixel resolution of 0.25 m. Considering the Swiss alpine region of about 20 000 km^2, performing the prediction and creating these shape files could take up to 30 days in this estimation. However, the task can be parallelised efficiently, reducing the production time significantly.

4.7 CONCLUSION

While OBIA is the state-of-the-art approach for mapping objects on remotely sensed images, it suffers from limitations that render this approach unsuitable for larger-scale studies. High-quality segmentation results come at the expense of a lack of transferability of parameter settings from one input image to another, manual adjustments, and a need for expert knowledge in applying the method to the specific task which together lead to long processing times. In particular, the first aspect generally hinders OBIA in a predictive setting for new images. To overcome these shortcomings and enable large-scale analysis, we compared OBIA to a fully-convolutional neural network approach which learns relevant features for segmentation by itself and thereby emulates some of the expert knowledge necessary to apply OBIA. We demonstrated that the U-Net approach is capable of performing as well as OBIA with respect to identifying trends in the spatial and temporal development of degraded soil, and can therefore replace OBIA in large-scale studies. Spatial patterns and temporal trends of both methods agree well; nevertheless, some generated segmentation results might partially not overlap ($F_1 = 78\%$ in the single-valley analysis). Specifically, we show that the U-Net (threshold 0.3) provides relative increases of total degraded area in the Urseren valley (201% in the single-valley and 132% multi-valley analysis) which match the estimates of OBIA (167%). This novel approach allows for individual threshold choices for the most successful representation of ongoing soil erosion processes. This is typically possible if some prior knowledge about erosion processes and the spatial extent of degraded soil is available, or if visual assessment is feasible, to which probability thresholds can be calibrated. In our study, we made use of training labels generated with OBIA. However, any kind of (high-)quality training labels can be used, and the U-Net erosion site segmentation is not limited to combined use with OBIA.

In summary, we show that with our approach we can perform erosion site prediction close to similar approaches such as OBIA which provide accurate segmentation results on small scales. A particular strength of the proposed approach is that similar trends are achieved with a more efficient, automatic, and objective method for mapping erosion sites. We require the U-Net approach to be trained only once and obtain much better transferability of the method to new images. Moreover, the approach is insensitive to the threshold choice with respect to trends of aggregated measures, and the improved running times make large-scale analysis of soil erosion is Swiss alpine grasslands feasible.

Still, our model is only as good as the training data; i.e. high-quality training data are important for adequate U-Net performance. Applications should include a variety of different sample regions to incorporate relevant erosion-type-specific conditions during training, like the orientation of erosion sites in the multi-valley analysis. Furthermore, the U-Net can use as many layers of information as required. A unique feature of fully-convolutional neural networks is that inputs of any size and any number of channels can be used, i.e. RGB images with DTM derivatives. Additional maps can be easily incorporated (see Figure 4.5), which might include more information, such as environmental properties or images with additional spectral information. In that regard, the U-Net has the advantage of continual learning; i.e. it can be trained further to incorporate conditions of completely new regions and erosion-type-specific properties. Generally, the U-Net model can be employed in a similar fashion for other segmentation tasks in remote sensing and other inputs, such as UAV or satellite imagery. The requirement for the input data is that the spatial resolution allows for identifying the target objects well enough.

5 INFORMED LATENT SPACE ENCODING THROUGH SIDE INFORMATION

Identifying meaningful and independent factors of variation in a dataset is a common but challenging learning task which is frequently addressed by means of deep latent variable models. This task can be viewed as learning symmetries preserving the value of a chosen property along latent dimensions. Such a target property can be, for instance, the orientation of an object in space or a chemical property of a compound. With a generative model we try to learn latent dimensions which allow separating property from other object information as much as possible. However, existing approaches exhibit severe drawbacks in enforcing this property *invariance* in the latent space. We address these shortcomings with a novel approach to cycle consistency. Our method involves two separate latent subspaces, one for the target property and the other for the remaining object information. In order to enforce invariance as well as sparsity which facilitates interpretation of the latent space, we incorporate semantic knowledge by using cycle consistency constraints relying on property side information. The proposed method is based on the deep variational information bottleneck and, in contrast to other approaches, allows using continuous target properties and provides inherent model selection capabilities. In this chapter, we demonstrate on synthetic and molecular data that our approach identifies more meaningful factors which lead to sparser and more interpretable models with improved property invariance. Furthermore, we showcase the disentanglement capability of our approach on the dSprites benchmark dataset.

5.1 INVARIANCES IN LATENT ENCODING

Understanding the nature of a generative process that provides us our observed data typically involves uncovering explanatory factors of variation which are responsible for the observations. But the relationship between these factors and our observation usually remains unclear. A common as-

Part of this chapter has been published in Samarin et al. (2021).

sumption is that the relevant factors can be expressed by a low-dimensional latent representation Z (Locatello et al., 2019). Therefore, popular machine learning methods involve learning appropriate latent representations to *disentangle* factors of variation. Learning disentangled representations is often considered in an unsupervised setting which does not rely on the prior knowledge about the data such as labels (Chen et al., 2016; Higgins et al., 2017; Chen et al., 2018b; Kim and Mnih, 2018; Lin et al., 2020). However, it was shown that an inductive bias on the dataset and learning approach is necessary to obtain disentanglement (Locatello et al., 2019). Inductive biases allow us to express assumptions about the generative process and to prioritise different solutions not only in terms of disentanglement (Higgins et al., 2017; Bouchacourt et al., 2018; Klys et al., 2018; Robert et al., 2019; Wieser et al., 2020), but also in terms of constrained latent space encodings (Keller et al., 2019, 2021), preservation of causal relationships (Wieczorek and Roth, 2016), or interpretability (Wu et al., 2017).

We consider a supervised setting where semantic knowledge about the input data allows structuring the latent representation in disjoint subspaces Z_0 and Z_1 of the latent space Z by enforcing *conditional invariance*. With this we mean that conditioning on part of the latent space, i.e. Z_0, allows property-invariant sampling in the latent subspace Z_1. In such supervised settings, disentanglement can be viewed as an extraction of level sets or symmetries inherent to our data X which leave a specified property Y invariant. An important application in that direction is the generation of diverse molecular structures with similar chemical properties (Wieser et al., 2020). The goal is to disentangle factors of variation relevant for the property. Typically, level sets L_y are defined implicitly through

$$L_y(f) = \{(x_1, ..., x_{d_x}) \mid f(x_1, ..., x_{d_x}) = y\} \tag{5.1}$$

for a property y which implicitly describes the level curve or surface with respect to inputs $[x_1, ..., x_{d_x}]^\top \in \mathbb{R}^{d_x}$. The topic of this chapter is to identify a sparse parametrisation of level sets which encodes conditional invariances and thus selects a correct model. Several techniques have been developed to steer model selection by sparsifying the number of features, e.g. Tibshirani (1996); Tishby et al. (1999), or compressing features into a low-dimensional feature space, e.g. Rey et al. (2014); Alemi et al. (2017); Wieczorek et al. (2018). These methods improve generalisation by focusing on only a subset of relevant features and using these to explain a phenomenon. Existing methods for including such prior knowledge in the model usually do not include dimensionality reduction techniques and perform a hand-tuned selection (Klys et al., 2018; Wieser et al., 2020; Keller et al., 2021).

In this chapter, we introduce a novel approach to cycle consistency – relying on property side information Y as our semantic knowledge – in order to provide conditional invariance in the latent space. By ensuring that our method fulfils cycle consistency and provides similar results for generated samples when fed back to the network, we achieve more disentangled and sparser representations. Our work builds on Wieczorek et al. (2018), where a general sparsity constraint on latent representations is provided, and on Klys et al. (2018); Wieser et al. (2020), where conditional invariance is obtained through adversarial training. We show that our approach addresses some drawbacks in previous approaches and allows us to identify more meaningful factors for learning better models and achieve improved invariance performance. Our contributions may thus be summarised as follows:

- We propose a novel approach for supervised disentanglement where conditional invariance is enforced by a novel cycle consistency constraint on property side information. This facilitates the guided exploration of the latent space and improves sampling with a fixed property.
- Our model inherently favours sparse solutions, leading to more interpretable latent dimensions and facilitates built-in model selection.
- We demonstrate that our method improves on the state-of-the-art performance for conditional invariance as compared to existing approaches on both synthetic and molecular benchmark datasets.

5.2 RELATED WORK

5.2.1 *Deep Latent Variable Models and Disentanglement*

Because of its flexibility, the variational autoencoder (VAE) (Kingma and Welling, 2014; Rezende et al., 2014) is a popular deep latent variable model in many areas such as fairness (Louizos et al., 2016), causality (Louizos et al., 2017), semi-supervised learning (Kingma et al., 2014), and design and discovery of novel molecular structures (Kusner et al., 2017; Gómez-Bombarelli et al., 2018; Nesterov et al., 2020). The VAE is closely related to the Information Bottleneck (IB) principle (Alemi et al., 2017; Tishby et al., 1999) as discussed in Section 3.5.3. Various approaches exploit this relation like the deep variational information bottleneck (DVIB) (Achille and Soatto, 2018; Alemi et al., 2017). Further extensions were proposed in the context of causality (Chicharro et al., 2020; Parbhoo et al., 2020a,b) or archetypal analysis (Keller et al., 2019, 2021).

The β-VAE (Higgins et al., 2017) extends the standard VAE approach and allows unsupervised disentanglement. In unsupervised settings, there exists a great variety of approaches based on VAEs and generative adversarial networks (GANs) to achieve disentanglement such as FactorVAE (Kim and Mnih, 2018), β-TCVAE (Chen et al., 2018b) or InfoGAN (Chen et al., 2016; Lin et al., 2020). Partitioning the latent space into subspaces is inspired by the multi-level VAE (Bouchacourt et al., 2018), where the latent space is decomposed into a local feature space that is only relevant for a subgroup and a global feature space. In supervised settings, several approaches such as Lample et al. (2017); Creswell et al. (2018); Klys et al. (2018); Wieser et al. (2020) achieve disentanglement by applying adversarial information elimination to select a model with partitioned feature and property space. In such a setting – and different to unsupervised disentanglement – our goal is supervised disentanglement with respect to a particular target property.

Another important line of research employs the idea of cycle consistency for learning disentangled representations. Presumably the most closely related work to this study is conducted by Jha et al. (2018); Wieser (2020). There, the authors employ a cycle-consistent loss on the latent representations to learn symmetries and disentangled representations in weakly supervised settings, respectively. Moreover, in Wieser et al. (2020), the authors use adversarial training and mutual information estimation to learn symmetry transformations instead of explicitly modelling them. In contrast, our work replaces adversarial training by using cycle consistency.

5.2.2 *Model Selection via Sparsity*

Several works perform model selection by introducing sparsity constraints which penalise the model complexity. A common sparsity constraint is the Least Absolute Shrinkage and Selection Operator (LASSO) (Tibshirani, 1996). Extensions of the LASSO propose a log-penalty to obtain even sparser solutions in the compressed IB setting (Rey et al., 2014) and generalise it further to deep generative models (Wieczorek et al., 2018). Furthermore, the LASSO has been extended to the group LASSO, where combinations of covariates are set to zero, the sparse group LASSO (Simon et al., 2013), and the Bayesian group LASSO (Raman et al., 2009). Perhaps most closely related to our work is the oi-VAE (Ainsworth et al., 2018), which incorporates a group LASSO prior in deep latent variable models. These methods employ a general sparsity constraint to achieve a sparse representation. Our model extends these ideas and imposes a semantic sparsity constraint in the form of cycle consistency that performs regularisation based on prior knowledge.

5.3 PRELIMINARIES

5.3.1 *Deep Variational Information Bottleneck*

We focus on the DVIB (Alemi et al., 2017) which is a method for information compression based on the IB principle (Tishby et al., 1999). We have introduced the DVIB in Section 3.5.3 and provide the main statements for this chapter in the following.

The objective is to compress a random variable X into a random variable Z while being able to predict a third random variable Y. The DVIB is closely related to the VAE (Kingma and Welling, 2014; Rezende et al., 2014). The optimal compression is achieved by solving the parametric problem

$$\min_{\boldsymbol{\phi},\boldsymbol{\theta}} I_{\boldsymbol{\phi}}(X;Z) - \lambda I_{\boldsymbol{\phi},\boldsymbol{\theta}}(Z;Y), \tag{5.2}$$

where I is the mutual information between two random variables (see Definition 6). Hence, the DVIB objective balances maximisation of $I_{\boldsymbol{\phi},\boldsymbol{\theta}}(Z;Y)$, i.e. Z being informative about Y, and minimisation of $I_{\boldsymbol{\phi}}(X;Z)$, i.e. compression of X into Z. We assume a parametric form of the conditionals $p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$[1] and $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{z})$ with $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ representing the parameters of the encoder and decoder network, respectively. Parameter λ controls the degree of compression and is closely related to β in the β-VAE (Higgins et al., 2017). The relationship to the VAE becomes more apparent with the definition of the mutual information terms:

$$I_{\boldsymbol{\phi}}(X;Z) = \mathbb{E}_{p(\boldsymbol{x})}\left[D_{KL}(p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z}))\right], \tag{5.3}$$

$$I_{\boldsymbol{\phi},\boldsymbol{\theta}}(Z;Y) \geqslant \mathbb{E}_{p(\boldsymbol{x},\boldsymbol{y})}\left[\mathbb{E}_{p_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{z})\right]\right] + h(Y), \tag{5.4}$$

with D_{KL} being the Kullback-Leibler divergence (see Definition 5), and $h(Y)$ the entropy.

1 Note that we adopt the notation as in Section 3.5. We indicate the probability density by $p(X = x) \equiv p(x)$, dropping the random variables in the notation.

5.3.2 *Cycle Consistency*

We use the notion of cycle consistency similar to Jha et al. (2018); Zhu et al. (2017). The CycleGAN (Zhu et al., 2017) performs unsupervised image-to-image translation, where a data point is mapped to its initial position after being transferred to a different space. For instance, suppose that domain X consists of images of summer landscapes, while domain Y consists of images of winter landscapes (see Figure 5.1). Function $f : x \mapsto y$ is used to transform a summer landscape $x \in X$ to a corresponding winter landscape $y \in Y$. Similarly, function $g : y \mapsto x$ maps y back to the domain X. The goal of cycle consistency is to learn a mapping g such that to $g(y) = \hat{x}$ is close to the initial x and thus $g \circ f$ is approximately the identity. The discrepancy between x and $\hat{x}$ is referred to as the *cycle consistency loss*. We minimise the loss $\|g(f(x)) - x\|_1$ to obtain an almost invertible mapping.

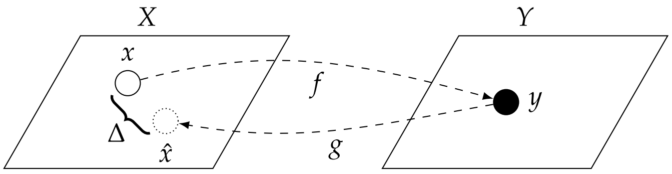


Figure 5.1: Illustration of the cycle consistency idea. Mappings $f : X \to Y$ and $g : Y \to X$ are learned such that $g \circ f$ is approximately the identity. The distance Δ between $x \in X$ and $g(f(x)) = \hat{x}$ is minimised through the cycle consistency loss.

5.4 METHODOLOGY

Our model is based on the DVIB to learn a compact latent representation, i.e. an encoder-decoder architecture. The input X and the output Y may be complex objects and can take continuous values. As a relevant example, we consider molecules with their respective molecular properties in this chapter. Unlike the standard DVIB, we do not only want to predict Y from an input X, but also want to generate new $\tilde{X}$ by sampling from our latent representation. As a consequence, we add an additional second decoder that reconstructs X from Z (similar to Gómez-Bombarelli et al. (2018) for decoder Y in the VAE setting), leading to the adjusted parametric objective

$$\min_{\boldsymbol{\phi},\boldsymbol{\theta},\boldsymbol{\tau}} I_{\boldsymbol{\phi}}(X;Z) - \lambda\big(I_{\boldsymbol{\phi},\boldsymbol{\theta}}(Z;Y) + I_{\boldsymbol{\phi},\boldsymbol{\tau}}(X;Z)\big), \tag{5.5}$$

where $\boldsymbol{\phi}$ are the encoder parameters, and $\boldsymbol{\theta}$ and $\boldsymbol{\tau}$ describe network parameters for decoding Y and X, respectively.

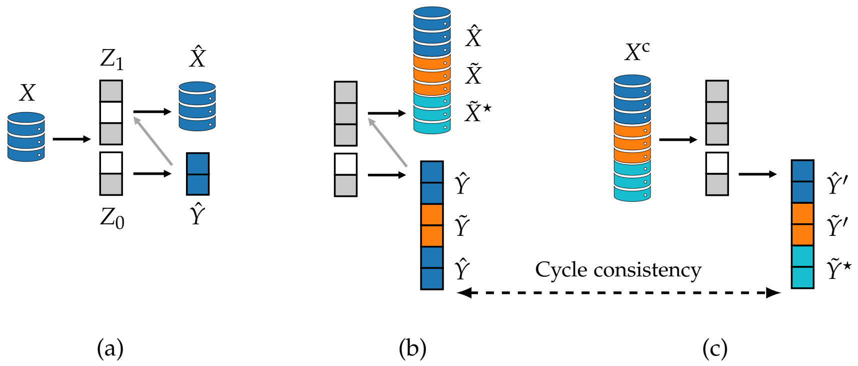


Figure 5.2: Overview of the proposed approach. (a) Firstly, we learn a sparse representation Z from our input data X which we separate into a property space Z_0 and an invariant space Z_1. Grey boxes indicate the (sparsely) selected latent dimensions. Given this representation, we try to predict the property $\hat{Y}$ and reconstruct our input $\hat{X}$. Grey arrows indicate that $\hat{Y} = \text{dec}_Y(Z_0)$ instead of Z_0 is used for decoding $\hat{X}$ (see Section 5.4.4). (b) Secondly, we sample new data in two ways: (i) uniformly in Z to get new data points $\tilde{X}$ and $\tilde{Y}$ (orange data), (ii) uniformly in Z_1 with fixed Z_0 to get $\tilde{X}^\star$ at fixed $\hat{Y}$ (cyan data). We concatenate the respective decoder outputs. (c) Lastly, we feed the concatenated input batch $X^c = (\hat{X}, \tilde{X}, \tilde{X}^\star)$ into our model and calculate the cycle consistency loss $\|\hat{Y} - \hat{Y}'\|_2 + \|\tilde{Y} - \tilde{Y}'\|_2 + \|\hat{Y} - \tilde{Y}^\star\|_2$ between the properties.

5.4.1 *Learning a Compact Representation*

Formulating our model as a DVIB allows leveraging properties of the mutual information with respect to learning compact latent representations. To see this, first assume that X and Y are jointly Gaussian distributed which leads to the *Gaussian Information Bottleneck* (Chechik et al., 2005) which we introduced in Section 3.5.3. The solution Z can be found analytically and proved to be Gaussian. In particular, for $X \sim \mathcal{N}(0, \Sigma_X)$, the optimal Z is a noisy projection of X: $Z = AX + \xi$, where $\xi \sim \mathcal{N}(0, \mathbb{1})$. The mutual information between X and Z is then equal to

$$I(X;Z) = \tfrac{1}{2} \log |A\Sigma_X A^\top + \mathbb{1}|, \tag{5.6}$$

where $|\cdot|$ denotes the determinant of the matrix. If we now assume A to be diagonal, the model becomes sparse (Rey et al., 2014). This is because a full-rank projection AX' of X' does not change the mutual information since $I(X;X') = I(X;AX')$. A reduction in mutual information can only be achieved by a rank-deficient matrix A. In general, the conditionals $Z|X$ and $Y|Z$ in Equation (5.2) may be parametrised by neural networks with X and Z as input. The diagonality constraint on A does not cause any loss of generality of the DVIB solution as long as the neural network encoder $\boldsymbol{f_\phi}$ makes it possible to diagonalise $A\boldsymbol{f_\phi}(X)\boldsymbol{f_\phi}(X)^\top A^\top$ (see Wieczorek et al. (2018) for more details). In the following, we consider A to be diagonal and define the sparse representation as the dimensions of the latent space Z selected by the non-zero entries of A. Recalling Equation (5.6), this allows us to approximate the mutual information for the encoder in Equation (5.3) in a sparse manner

$$I_\phi(X;Z) = \tfrac{1}{2}\log|\mathrm{diag}(\boldsymbol{f_\phi}(X)\boldsymbol{f_\phi}(X)^\top) + \mathbf{1}|, \tag{5.7}$$

where $\mathbf{1}$ is the all-one vector and the diagonal elements of A are subsumed in the encoder $\boldsymbol{f_\phi}$.

5.4.2 *Conditional Invariance and Informed Sparsity*

A general sparsity constraint is not sufficient to ensure that latent dimensions indeed represent independent factors. In a supervised setting, our target Y conveys semantic knowledge about the input X, e.g. a chemical property of a molecule. To incorporate semantic knowledge into our model, we require a mechanism that partitions the representation such that it encodes the semantic meaning not only sparsely but preferably independently of other information concerning the input.

To this end, the central element of our approach is cycle consistency with respect to target property Y, which is illustrated in steps (b) and (c) in Figure 5.2. The idea is that reconstructed $\hat{X}$ or newly sampled $\tilde{X}$ with associated prediction $\hat{Y}$ and $\tilde{Y}$ are expected to provide matching predictions $\hat{Y}'$ and $\tilde{Y}'$ when $\hat{X}$ and $\tilde{X}$ are used as an input to the network. This means, if we perform another cycle through the network with sampled or reconstructed inputs, the property prediction should stay consistent. The partitioning of the latent space Z in the property subspace Z_0 and the *invariant* subspace Z_1 is crucial. The property Y is predicted from Z_0, while the input is reconstructed from the full latent space Z. Ensuring cycle consistency with respect to the property allows putting property-relevant

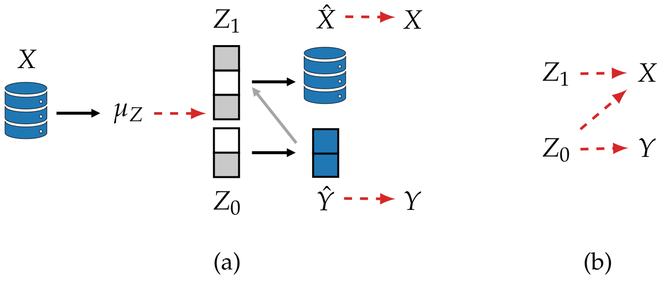


Figure 5.3: Probabilistic graph of our model. (a) Dashed red arrows indicate probabilistic links: (1) μ_Z is a deterministic transformation of X, while $Z = (Z_0, Z_1)$ is sampled from $\mathcal{N}(\mu_Z, \sigma^2_{\text{noise}})$ (see Section 5.4.1 and 5.4.4); (2) the mean squared error loss corresponds to the negative log-likelihood of a Gaussian distribution (see Section 3.1.2) where $\hat{X}$ is the (unbiased) estimate for the mean with (noisy) sample X (similarly for $\hat{Y}$ and Y). (b) Probabilistic relationship between random variables Z_0, Z_1, X, and Y. Fixing Z_0 renders X and Y conditionally independent, i.e. $X \perp\!\!\!\perp Y \mid Z_0$.

information into the property subspace Z_0. Furthermore, the latent space is regularised by drawing samples which adhere to cycle consistency and provide additional sparsity. If information about Y is encoded in Z_1, this will lead to a higher cycle consistency loss. In this way, cycle consistency enforces invariance in subspace Z_1. By fixing coordinates in Z_0, and thus fixing a property, sampling in Z_1 results in newly generated $\tilde{X}$ with the same property $\tilde{Y}$. More formally, fixing Z_0 renders random variables X and Y conditionally independent, i.e. $X \perp\!\!\!\perp Y \mid Z_0$ (see Figure 5.3). We ensure conditional invariance with a particular sampling scheme: We fix the Z_0 coordinates and sample in Z_1 to obtain generated $\tilde{X}^\star$ all with a fixed property $\hat{Y}$. Using these inputs allows to obtain a new prediction $\tilde{Y}^\star$ which should be close to the fixed target property $\hat{Y}$. We choose the ℓ_2-norm for convenience and define the full cycle consistency loss by

$$J_{\text{cycle}} = \|\hat{Y} - \hat{Y}'\|_2 + \|\tilde{Y} - \tilde{Y}'\|_2 + \|\hat{Y} - \tilde{Y}^\star\|_2. \tag{5.8}$$

5.4.3 *Proposed Framework*

The resulting model combines sparse DVIBs with partitioned latent space and a novel approach to cycle consistency, which drives conditional invariance and informed sparsity in the latent encoding. This allows latent dimensions in Z_0 relevant for prediction of Y to disentangle from latent dimensions in Z_1 which encode remaining input information of X. The objective function is given by

$$J(\boldsymbol{\phi}, \boldsymbol{\theta}, \boldsymbol{\tau}) = I_{\boldsymbol{\phi}}(X; Z) - \lambda\big(I_{\boldsymbol{\phi},\boldsymbol{\tau}}(Z_0, Z_1; X) + I_{\boldsymbol{\phi},\boldsymbol{\theta}}(Z_0; Y) - \beta J_{\text{cycle}}\big). \tag{5.9}$$

The proposed model performs model selection as it inherently favours sparser latent representations. This in turn facilitates easier interpretation of latent factors because of the built-in conditional independence between property space Z_0 and invariant space Z_1. These adjustments address some of the issues of the STIB (Wieser et al., 2020) relying on adversarial training, mutual information estimation (which can be difficult in high-dimensions (Song and Ermon, 2020)) and bijective mapping which can make the training challenging. In contrast to the work of Jha et al. (2018); Wieser (2020), we use a general sparsity constraint on the encoder (see Section 5.4.1) and impose a novel cycle consistency loss on the predicted outputs Y instead of the latent representation Z. A reason to consider Y rather than Z is that varying latent dimensionality leads to severe problems in the optimisation process as it requires an adaptive rescaling of the different loss weights. To overcome this drawback, we close the full cycle and define the loss on the outputs.

5.4.4 *Implementation*

In Algorithm 1, we provide more details on the implementation with the individual steps being depicted in Figure 5.2. As outlined in Section 3.5.3, we use a variational inference approach to approximate the true posterior distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ by $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$, as it is common for VAEs. Furthermore, we only learn the mean $\boldsymbol{\mu}_Z$ of the latent representation. We optimise the variance $\boldsymbol{\sigma}_Z^2$ as a free parameter or fix the variance to 1, but do not learn the variance from data. For this reason, we will refer to $\boldsymbol{\sigma}_Z^2 \equiv \boldsymbol{\sigma}_{\text{noise}}^2$ as sampling noise.

In the first part of our training algorithm, we map the input X to the latent representation Z (l. 5) which is partitioned into two subspaces Z_0 and Z_1 (l. 6). From Z_0 we make a prediction for Y (l. 7). We choose to concatenate the decoded Z_0 (instead of just Z_0) with Z_1 in order to decode $\hat{X}$, i.e. $\hat{X} = \text{dec}_X(Z_1, \hat{Y} = \text{dec}_Y(Z_0))$. This is an additional measure to ensure that Z_0 contains information relevant for property prediction Y and prevents superfluous remaining information about the input X in property space Z_0. We use Gaussian decoders (see Equation (3.78)). Thus, both reconstruction and prediction loss (terms $I_{\boldsymbol{\phi},\boldsymbol{\tau}}$ and $I_{\boldsymbol{\phi},\boldsymbol{\theta}}$ in Equation (5.9)) are approximated by MSE losses. In the second part of our method, we draw samples in two ways (ll. 12 – 14): (i) uniformly in Z to obtain $\tilde{Y}$ and $\tilde{X}$ and (ii) uniformly in Z_1 with fixed Z_0 to get $\tilde{X}^\star$ at a fixed $\hat{Y}$. The samples are drawn from a uniform distribution within the interval of minus two sample standard deviations to plus two sample standard deviations (in the respective latent dimension). We chose uniform sampling to regularise the latent dimensions evenly. In the third step, we concatenate the reconstructed and sampled data to a new input $X^c = (\hat{X}, \tilde{X}, \tilde{X}^\star)$ and perform a second cycle through our model (l. 17). Using the new encoding Z_0^c allows to decode property predictions $\hat{Y}_i', \tilde{Y}_i', \tilde{Y}_i^\star$ (l. 19) and calculate the cycle consistency loss according to Equation (5.8) (l. 20). All required expectations are approximated with Monte Carlo estimates. We update the model parameters $\boldsymbol{\phi}, \boldsymbol{\theta}$, and $\boldsymbol{\tau}$ by taking a gradient step using the loss function Equation (5.9). Lastly, we increase the compression parameter λ by an annealing factor l (l. 24; predefined by hyperparameter tuning) after every epoch, similar to learning rate scheduling. The described algorithm can be trained with any kind of gradient descent method until convergence. A TensorFlow (Abadi et al., 2015) implementation is available under the GNU General Public License v3.0.[2]

5.5 EXPERIMENTAL EVALUATION

We evaluate the effectiveness of our proposed method with respect to (i) selection of a sparse representation with meaningful factors of variation (i.e. model selection) and (ii) enforcing conditional independence in the latent space between these factors. To this end, we conduct experiments on a synthetic dataset with knowledge about appropriate parametrisations to highlight the differences to existing models. Additionally, we evaluate our model on a real-world application and a standard disentanglement benchmark with a focus on conditional invariance and generation of novel samples. To assess the performance of our model, we compare our approach to two state-of-the-art baselines: (i) the β-VAE (Higgins et al., 2017) which is

2 https://github.com/bmda-unibas/CondInvarianceCC

Algorithm 1 Pseudocode of our method employing property cycle consistency for conditional invariance.

```
Input: input $x$, target $y$
 1: for each epoch do
 2:     sample $i$ mini-batches of $x$ and $y$
 3:     for each mini-batch $i$ do
 4:         Part 1
 5:         encode $x_i$ into $z_i \sim q_\phi(z_i \mid x_i)$
 6:         split $z_i$ in $z_0$ and $z_1$
 7:         decode $z_0$ to obtain $\hat{y}_i \sim p_\theta(\hat{y}_i \mid z_0)$
 8:         decode $(\hat{y}_i, z_1)$ to obtain $\hat{x}_i \sim p_\tau(\hat{x}_i \mid \hat{y}_i, z_1)$
 9:         calculate reconstruction and prediction losses
10:
11:         Part 2: Sampling
12:         sample (i) $\tilde{z}$ and (ii) $\tilde{z}_1^\star$ randomly from uniform prior
            $\mathcal{U}(-2 \cdot \sigma_{\text{signal}}, 2 \cdot \sigma_{\text{signal}})$ in (i) full $z$ and (ii) only in $z_1$, resp.
13:         decode $\tilde{z}_0$ to obtain $\tilde{y}_i \sim p_\theta(\tilde{y}_i \mid \tilde{z}_0)$
14:         decode $(\tilde{y}_i, \tilde{z}_1)$, $(\hat{y}_i, \tilde{z}_1^\star)$ to obtain $\tilde{x}_i \sim p_\tau(\hat{x}_i \mid \tilde{y}_i, \tilde{z}_1)$,
            $\tilde{x}_i^\star \sim p_\tau(\hat{x}_i^\star \mid \hat{y}_i, \tilde{z}_1^\star)$
15:
16:         Part 3: Cycle consistency
17:         encode $x^c = (\hat{x}_i, \tilde{x}_i, \tilde{x}_i^\star)$ into $z_i^c \sim q_\phi(z_i^c \mid \hat{x}_i, \tilde{x}_i, \tilde{x}_i^\star)$
18:         split $z_i^c$ in $z_0^c$ and $z_1^c$
19:         decode $z_0^c$ to obtain $\hat{y}_i', \tilde{y}_i', \tilde{y}_i^\star \sim p_\theta(\hat{y}_i', \tilde{y}_i', \tilde{y}_i^\star \mid z_0^c)$
20:         calculate cycle consistency loss:
            $||\hat{y}_i - \hat{y}_i'||_2 + ||\tilde{y}_i - \tilde{y}_i'||_2 + ||\hat{y}_i - \tilde{y}_i^\star||_2$
21:
22:         update $\phi, \theta, \tau$ by taking a gradient step
23:     end for
24:     increase $\lambda$ by annealing factor $l$
25: end for
```

a typical baseline model in disentanglement studies and (ii) the symmetry-transformation information bottleneck (STIB) (Wieser et al., 2020) which ensures conditional invariance through adversarial training and is the direct competitor to our model. We adapt the β-VAE by adding a decoder for property Y (similar to Gómez-Bombarelli et al. (2018)) which takes only subspace Z_0 as input. The latent space of the adapted β-VAE is split into two subspaces as in the STIB and our model, but has no explicit mechanisms to enforce invariance. This setup can be viewed as an ablation study in which the β-VAE is the basis model of our approach without cycle consistency and sparsity constraints. The STIB provides an alternative approach for the same goal but with a different mechanism.

The objective of the supervised disentanglement approach is to ensure disentanglement of a fixed property with respect to variations in the invariant space Z_1. This is a slightly different setting than in standard unsupervised disentanglement and therefore standard disentanglement metrics might be less insightful and are not considered. Instead, in order to test the property invariance, we first encode the inputs of the test set and fix the coordinates in the property subspace Z_0 which provides prediction $\hat{Y}$. Then we sample uniformly at random in Z_1 (plus/minus one standard deviation), decode the generated $\tilde{X}$ and perform a cycle through the network to obtain $\tilde{Y}$. This provides the predicted property for the generated $\tilde{X}$. If conditional invariance between X and Y at a fixed Z_0 is warranted, the mean absolute error (MAE) between $\hat{Y}$ and $\tilde{Y}$ should be close to zero. Thus, all models are trained to attain similar MAEs for reconstructing X and, in particular, predicting Y, to ensure a fair comparison.

5.5.1 *Synthetic Dataset*

In the first experiments, we focus on learning level sets of ellipses ($d_x = 2$) and ellipsoids ($d_x = 3$) mapped into five dimensions ($d'_x = 5$). We consider these experiments as they allow a clear interpretation and visualisation of fixing a property, i.e. choosing the ellipse curve or ellipsoid surface, and known low-dimensional parametrisations are readily available. To this end, we sample uniformly at random data points X_{original} from $\mathcal{U}([-1,1]^{d_x})$ and calculate the corresponding one-dimensional properties Y_{original} ($d_y = 1$) with

$$y = \begin{cases} x_1^2/4 + x_2^2/2 & (d_x = 2), \\ x_1^2/4 + x_2^2/2 + x_3^2 & (d_x = 3). \end{cases} \tag{5.10}$$

We then rotate the ellipses / ellipsoids in the X_1X_2-plane by $\pi/4 = 45^\circ$ and add Gaussian noise $\epsilon \sim \mathcal{N}(0, 0.04)$. In Figure 5.4, we illustrate the two-dimensional case of the ellipse because it is easier to visualise. Equation (5.10) implicitly defines level sets, i.e. the ellipse curves or ellipsoid surfaces (see Figure 5.4a). Common (sparse) parametrisations consider polar coordinates

$$(x, y) = (r\cos\varphi, r\sin\varphi) \tag{5.11}$$

for the ellipse and spherical coordinates

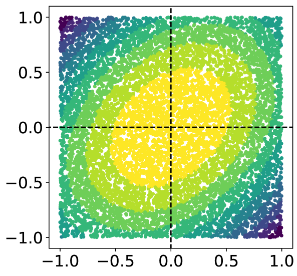
$$(x, y, z) = (r\cos\varphi\sin\vartheta, r\sin\varphi\sin\vartheta, r\cos\vartheta) \tag{5.12}$$

for the ellipsoid, with radius $r \in [0, \infty)$, (azimuth) angle $\varphi \in [0, 2\pi)$ in the X_1X_2-plane, and polar angle $\vartheta \in [0, \pi]$ measured from the X_3 axis.

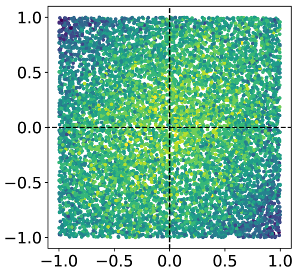
In a real-world scenario, we typically do not have access to the underlying data generating process providing X_{original} and the (noisy) property Y_{original} but a transformed view on these quantities. To reflect this, we map the input X_{original} into a five dimensional space, i.e. $X_{\text{original}} \in [-1, 1]^{N \times d_x} \to X \in \mathbb{R}^{N \times d'_x}$, and property Y_{original} into three dimensional space, i.e. $Y_{\text{original}} \in \mathbb{R}_+^{N \times d_y} \to Y \in \mathbb{R}^{N \times d'_y}$, with the number of samples N and dimensions $d_x = \{2, 3\}$, $d_y = 1$, $d'_x = 5$ and $d'_y = 3$. We do so by filling up the additional dimensions with random numbers sampled from the Gaussian $\mathcal{N}(0, 0.0001)$ and performing a (fixed) random rotation $R_X \in \mathrm{SO}(d'_x = 5)$ and $R_Y \in \mathrm{SO}(d'_y = 3)$ on the resulting matrices to obtain input X and property Y.

The goal of our experiment is to identify a low-dimensional parametrisation which captures the underlying radial and angular components, i.e. identify latent dimensions which correspond to parameters (r, φ) and (r, φ, ϑ). Note that for a particular level curve or surface the radius depends on the angles, i.e. $r(\varphi)$ or $r(\varphi, \vartheta)$, respectively. But for fixed angles, there is a unique relationship between the level set y and the radius r and thus for any angle. A representation of r allows reconstructing y and the angle representations allow generating the data points on the level curve / surface.

ARCHITECTURE AND TRAINING: For all models, the encoder and the input decoder consist of two fully-connected hidden layers. The property decoder consists of one fully-connected hidden layer. Each hidden layer has 256 units. The latent space is chosen to be eight-dimensional, $d_z = 8$, with three dimensions $d_{z_0} = 3$ reserved for property subspace Z_0 and five dimensions $d_{z_1} = 5$ for invariant subspace Z_1. We choose a generous



(a)

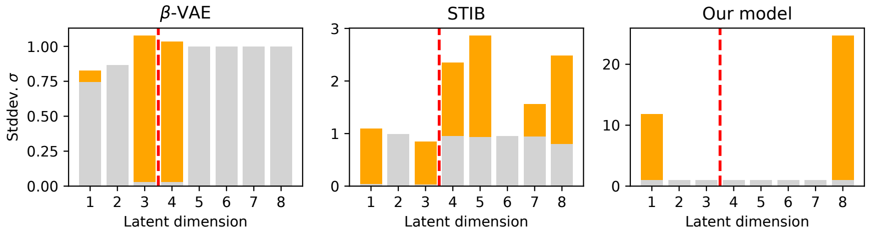


(b)

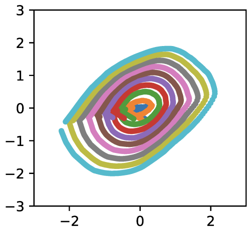
Figure 5.4: Input data for ellipse ($d_x = 2$) example. Data points ($N = 10\,000$) are uniformly sampled in $[-1, 1]^2$, properties Y are binned for illustration purposes with different colours indicating different level sets. Original input data (a) without noise and (b) with additional noise $\epsilon \sim \mathcal{N}(0, 0.04)$ on property Y (before mapping into higher dimension).

latent space size here with $d_{z_1} = d'_x = 5$ and $d_{z_0} = d'_y = 3$ to evaluate the sparsity and model selection. Choosing larger dimensions for the latent subspaces than in the data provides no additional benefit. This is because we expect the latent representation to be low-dimensional compared to the input and not higher-dimensional, typically. The (adapted) β-VAE and STIB are trained in the standard published way. Note that in our model, the noise level is fixed at $\sigma_{\text{noise}} = 1$ w.l.o.g. (see Section 5.4.1). For our model in the synthetic experiments, we add a small additional variance of 10 in Equation (5.7) in the last dimension, i.e. adjusting all-one vector $(1, ..., 1, 1)^\top \rightarrow (1, ..., 1, 10)^\top$. We found that this adaption does not drastically affect the learned parametrisation or sparsity but encodes the assumption about discontinuities, for instance, an angular parameter spanning $[0, 2\pi)$. This helps to reduce the gap of the discontinuity when illustrating the latent traversal. Training data is sampled from the generating process with an independent test dataset being fixed at the beginning of training. The model is trained with an Adam optimiser (Kingma and Ba, 2015a), a learning rate of 10^{-4} and a batch size of 60.

RESULTS: All models attain similar MAEs for X reconstruction and Y prediction but differ in the property invariance as summarised in Table 5.1. Our model learns more invariant representations with several factors difference with respect to the property invariance in both experiments. In Figure 5.5a, signal vs. noise for the different models is presented. The

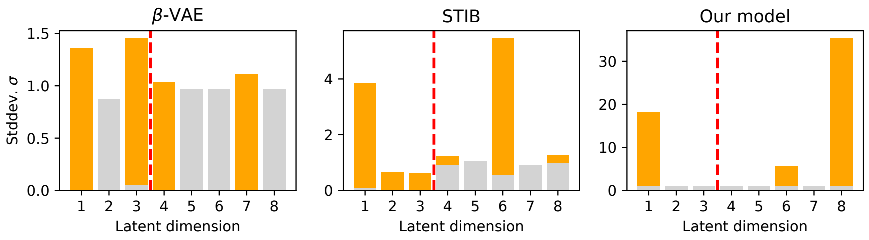


(a)

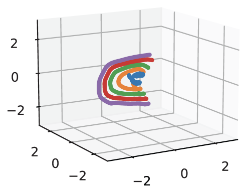


(b)

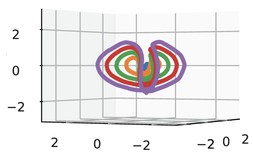
Figure 5.5: Results for ellipses in original input space ($d_x = 2$). (a) Illustration of standard deviation in the different latent dimensions, where property subspace Z_0 spans dimensions $1 - 3$ and invariant subspace Z_1 spans dimensions $4 - 8$. We consider a latent dimension to be selected if the signal exceeds the noise, i.e. orange bars are visible. Only our model selects the expected numbers of parameters. (b) Illustration of latent traversal in our model in latent dimension 8 in the original input space for fixed values in the property space dimension 1 (different colours). The selected dimension 8 represents the angular component φ and reconstructs the full ellipse curves. The latent traversals of all latent dimension 4 to 8 in our model can be found in Appendix Figure A.1.



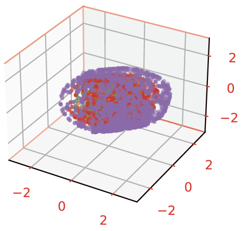
(a)



(b)



(c)



(d)

Figure 5.6: Results for ellipsoids in original input space ($d_x = 3$). (a) Illustration of standard deviation in the different latent dimensions, where property subspace Z_0 spans dimensions $1-3$ and invariant subspace Z_1 spans dimensions $4-8$. We consider a latent dimension to be selected if the signal exceeds the noise, i.e. orange bars are visible. Only our model selects the expected numbers of parameters. (b,c) Illustration of latent traversal in our model in latent dimensions 6 and 8 in the original input space for fixed values in the property space dimension 1 (different colours). The selected dimension 6 in (b) represents the polar angle ϑ, while dimension 8 in (c) can be related to the azimuth angle φ. (d) Samples in all selected dimensions (i.e. 6 and 8) reconstruct the full ellipsoid. We intentionally did not sample the ellipsoid surfaces completely to allow seeing surfaces underneath. The latent traversals of all latent dimension 4 to 8 in our model can be found in Appendix Figure A.2.

standard deviation σ_{signal} is calculated as the sample standard deviation of the learned means in the respective latent dimension. The sampling noise σ_{noise} is optimised as a free parameter during training. We consider a latent dimension to be informative or selected if the signal exceeds the noise. The sparsest solution is obtained in our model with one latent dimension selected in the property subspace Z_0 and one in the invariant subspace Z_1. In Figure 5.5b, we examine the obtained solution more closely in the original data space by mapping back from $d'_x = 5$ to $d_x = 2$ dimensions. We consider ten equidistant values in the selected Z_0 dimension 1 and sample points in the selected Z_1 dimension 8. The different colours represent fixed values in Z_0, with latent traversal in Z_1 dimension 8 reconstructing the full ellipse. This means, the selected latent dimension 8 contains all relevant information at a given coordinate in Z_0, while dimensions 4 to 7 do not contain any relevant information. See Appendix Figure A.1 for the latent traversals of all latent dimension 4 to 8. We can relate the selected dimension 1 in Z_0 to the radius r and dimension 8 in Z_1 to the angle φ.

For the ellipsoid ($d_x = 3$) in Figure 5.6a, we obtain qualitatively the same results as for the ellipse. Again, only our model selects the correct number of latent factors with one in Z_0 and two in Z_1. The latent traversal results are more intricate to interpret. For latent dimension 6, we obtain a representation which can be interpreted as encoding the polar angle ϑ (see Figure 5.6b). Traversal in latent dimension 8 (see Figure 5.6c) yields closed curves in three dimensions which can be viewed as on orthogonal representation to dimension 6 and be interpreted as an encoding of the azimuth angle φ. In Figure 5.6d, the last plot shows sampling in the selected Z_1 dimensions for fixed Z_0 (i.e. property Y) and reconstructs the full ellipsoid. Appendix Figure A.2 provides the latent traversal results for all latent dimension 4 to 8. Although β-VAE and STIB perform equally well on reconstructing and predicting on the test set, these models do not consistently lead to sparse and easily interpretable representations which allow direct traversal on the level sets as shown for our model. The presented results remain qualitatively the same for reruns of the models.

5.5.2 *Small Organic Molecules*

As a more real-world example, we consider generation of novel molecules with desirable properties in drug and material design. This task is challenging due to the large, unstructured, and discrete molecular space with the number of potentially drug-like molecules being estimated to range

Table 5.1: Mean absolute errors (MAE) for reconstruction of input X, prediction of property Y, and the property invariance on the test set. MAEs on the transformed 5-dimensional input X and the 3-dimensional property Y are provided.

	Ellipse			Ellipsoid		
Model	X	Y	Invar.	X	Y	Invar.
β-VAE	0.03	0.25	0.058	0.02	0.25	0.153
STIB	0.03	0.25	0.027	0.03	0.25	0.083
Ours	0.04	0.25	**0.006**	0.05	0.25	**0.006**

between 10^{23} to 10^{60} (Gómez-Bombarelli et al., 2018). A systematic synthesis and testing of novel compounds is not only resource but also time intensive and thus impractical. Therefore, an informed prescreening of candidate compounds with fixed properties, as presented through our method, can be a highly useful tool, potentially. To this end, we consider the QM9 dataset (Ramakrishnan et al., 2014) which includes 133 885 organic molecules. The molecules consist of up to nine heavy atoms (C, O, N, and F), not including hydrogen. Each molecule includes corresponding geometric, energetic, electronic and thermodynamic properties obtained from Density Functional Theory computations. In our experiments, we select a subset with a fixed stoichiometry ($C_7O_2H_{10}$) which consists of 6093 molecules. As the chemical property, we choose the band gap energy.

ARCHITECTURE AND TRAINING: For the input X we use the bag-of-bonds (Hansen et al., 2015) descriptor as a translation, rotation, and permutation invariant representation of molecules, which involves 190 dimensions. We standardise X and Y, i.e. we compute z-scores. For the encoder and decoder of our model, we employ two fully-connected hidden layers with 1024 units each. The property prediction network consists of two fully-connected hidden layers with 128 units each. For activation, a ReLU activation function is used. The number of latent dimensions is set to $d_z = 17$, where Z_0 consists of one dimension and Z_1 consists of the remaining 16 dimensions. To evaluate the property invariance, we first adjust the regularisation loss weights for a fair comparison of the models. The weights for the irrelevance loss in the STIB model and the invariance loss terms in our model were increased until a drop in reconstruction and prediction performances com-

Table 5.2: Mean absolute errors (MAE) for reconstruction of input X, prediction of property Y, and the property invariance on the test set. MAEs on property Y (band gap energy) and property invariance are given in kcal mol^{-1}.

	Molecules		
Model	X	Y	Invar.
β-VAE	0.01	4.01	5.66
STIB	0.01	4.08	3.05
Ours	0.01	4.06	**1.34**

pared to the β-VAE results was noticeable. The model is trained with an Adam optimiser, a learning rate of 10^{-4}, and a batch size of 250. Each model is trained on a 5750 training set and evaluated on a 300 out-of-sample set.

RESULTS: Table 5.2 summarises the results. On a test set of 300 molecules, all models achieve similar MAE of 0.01 for the reconstruction of X. For prediction of the band gap energies Y an MAE of approximately 4 kcal mol^{-1} is achieved. The invariance is computed on the basis of 25 test molecules and 400 samples generated for each reference molecule. Similarly to the synthetic experiments, the STIB model performs almost twice as well as the β-VAE, while our model yields a distinctly better invariance of 1.34 kcal mol^{-1} among both models. With this result, we can generate novel molecules which are very close to a fixed property. This capability is illustrated in Figure 5.7. For two reference molecules in the test set, we generate 2000 new molecules by sampling uniformly at random with one standard deviation in the invariant subspace Z_1 and keeping the reference property value, i.e. fixed Z_0 coordinates. We show three such examples in Figure 5.7a and select the nearest neighbours in the test set for visualisation of the molecular structure. For all samples, the box plots in Figure 5.7b illustrate the distribution in the predicted property values. The spread of predicted property values is generally smaller than the model prediction error of 4.06 kcal mol^{-1} and the predicted property of a majority of samples is close to the target property value.

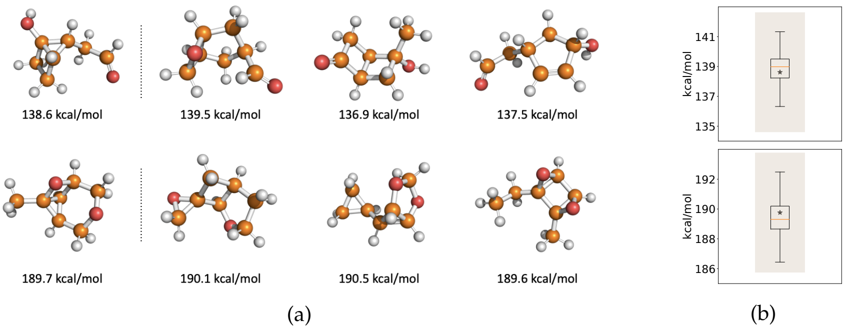


Figure 5.7: Illustration of the generative capability of our model for two reference molecules (rows). (a) The first molecule is the reference molecule with a fixed reference band gap energy. We display three samples and their predicted band gap energies out of 2000 samples. (b) Box plots for distribution of predicted property. The star symbol marks the fixed reference band gap energy. The shaded background depicts the prediction error range of the model.

5.5.3 *Disentanglement in dSprites*

Finally, we illustrate the disentanglement capability of our approach on the dSprites dataset (Matthey et al., 2017). The dataset comprises 737 280 images of three shapes (square, ellipse, heart) on six different scales, 40 orientations and 32×32 positions for coordinates (x, y). In the following, we consider six independent settings for the target property Y where we choose either the (i) x-position, (ii) y-position, (iii) shape, (iv) scale, (v) orientation, or (vi) jointly (x, y) position, scale, and orientation as the respective target property. In the last setting all factors except for the shape are used as target properties and we refer to this setting as *no shape*. In the settings involving the orientation as a property, we reduce rotations to the $[0, \pi/2)$ or $[0^\circ, 90^\circ)$ interval, which is a reduction of the dataset to a quarter or 184 320 images. This is necessary because rotating a square by $\pi/2 \equiv 90^\circ$ results in orientations indistinguishable from the original orientation without any rotation. Rotations exceeding this interval lead to squares which look exactly alike but have different orientation values.[3] Therefore, we omit these cases in our supervised problem setting, which would otherwise hinder the

3 The same holds true for ellipses and rotations exceeding the interval $[0, \pi)$ or $[0^\circ, 180^\circ)$. Thus, considering only rotations in $[0, \pi/2)$ or $[0^\circ, 90^\circ)$ also avoids ambiguity in orientation labels for ellipses, too.

training and thus generalisation capability of our approach. Furthermore, orientation and shape values are normalised to the $[0, 1]$ interval.

ARCHITECTURE AND TRAINING: The encoder and input decoder are defined analogous to the FactorVAE architecture for dSprites (Kim and Mnih, 2018). On the encoder side, the greyscale input images of 64×64 pixels are processed by four convolution layers with $\{32, 32, 64, 64\}$ feature maps, respectively. The resulting feature maps are then flattened and a fully-connected layer with d_z hidden units parametrises the means μ_Z in the latent space. On the input decoder side, the latent code Z is processed by two fully-connected layers with 128 and 1024 hidden units as well as four transposed convolution layers which perform up-sampling with $\{64, 32, 32, 1\}$ feature maps, respectively. In all (transposed) convolution layers, a filter size of 4×4 and strides of size 2 are used. In all layers, ReLU activations are employed except for the fully-connected layer mapping to the latent space and the last transposed convolution layer. Inspired by the β-VAE decoder for the dSprites dataset (Higgins et al., 2017), in the property decoder the latent code Z_0 is processed by three fully-connected layers with $\{1200, 1200, 4096\}$ hidden units and ReLU activations followed by a final fully-connected layer with sigmoid activation which maps to the property output. We choose the latent dimensionality to match the number of underlying factors of variation in dSprites, i.e. $d_z = 5$. In the different settings, the property subspace is of size $d_{z_0} = 1$ for the single target property settings (i) to (v) and $d_{z_0} = 4$ for the multi-target property setting (vi). With these choices the latent representation is already of the desirable sparsest size. The primary goal is to study disentanglement of latent dimensions and latent subspace Z_0 and Z_1 in our approach. As before, our model is trained with the Adam optimiser, a learning rate of 10^{-4}, and a batch size of 256. In these experiments, we only consider our approach and train the model for 2 000 000 iterations, reporting the results at the end of training. A random training and test set split of 80%/20% is used.

RESULTS: For contextualisation, we consider the mean model which predicts the mean image and mean property – obtained over the whole training set – for any kind of input image. Table 5.3 provides the quantitative results for all six settings of target properties. In summary, the (test) input reconstruction MAE is at least an order of magnitude and the (test) property prediction MAE at least one to two orders of magnitude lower than the prediction of the mean model. Generally, the MAEs for the

Table 5.3: Mean absolute errors (MAE) for reconstruction of input X, prediction of property Y, and the property invariance on the test set. Six different model settings are considered with different target properties. *No shape* indicates that all factors except the shape are used as target properties. For contextualisation, we consider the mean model which predicts the mean image and mean property – obtained over the whole training set – for any kind of input image.

	Our model			Mean model	
Property	X	Y	Invar.	X	Y
x	25	0.0033	0.0041	308	0.2581
y	21	0.0037	0.0036	308	0.2581
shape	30	0.0007	0.0010	308	0.2222
scale	28	0.0016	0.0021	308	0.1500
orientation	14	0.0013	0.0018	308	0.2778
no shape	25	0.0139	0.0118	308	0.9439

property invariance and model error for predicting the property match. This is in contrast to the previous experiments where the model prediction error exceeded the property invariance (see Tables 5.1 and 5.2). However, by design of the dataset, properties can be disentangled from other factors and thus low property prediction errors are attainable which limit the errors for the property invariance.

More qualitatively, we observe disentanglement between latent subspaces Z_0 and Z_1. Figure 5.8 shows for different sprites a latent traversal in property subspace Z_0 in the different target property settings. We select sprites of the test set, fix the latent coordinates in the invariant subspace Z_1 and change latent coordinates in property subspace Z_0, enabling us to generate (artificial) samples with different values for the property. For the one target property settings, values are changed from small (negative) to large (positive) values in the respective latent dimension. In the setting with four properties, one of the latent dimensions in subspace Z_0 was selected and values were chosen from small (negative) to large (positive) values. While the target property varies, as for example the position of the sprite in the image or its orientation, all other factors remain conserved. In the *no shape* setting, the latent traversal indeed changes all property factors, i.e. (x, y) coordinates, scale, and orientation, while the shape is unchanged.

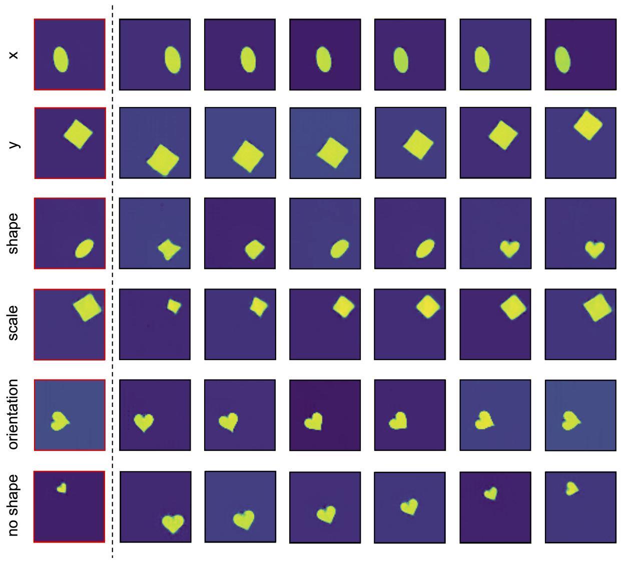


Figure 5.8: Latent traversal in property subspace Z_0. Different sprites of the test set (reconstruction results outlined in red; first column) are selected and several models with different target properties (rows) are consider: (i) x-position, (ii) y-position, (iii) shape, (iv) scale, (v) orientation, and (vi) jointly (x, y) position, scale, and orientation (*no shape*). The property value is varied from small (negative) to large (positive) values in the latent subspace Z_0 at fixed latent coordinates in the invariant subspace Z_1 and the corresponding samples are generated (second to last column). The model allows varying the target property while leaving the other factors of variation unchanged, as is illustrated for the selected sprites.

The opposite case is considered in Figure 5.9 with a latent traversal in the invariant subspace Z_1. For sprites of the test set, the latent coordinates in the property subspace Z_0 are fixed – corresponding to fixed property values – and latent coordinates in the invariant subspace Z_1 are sampled uniformly at random. The generated samples conserve the property, e.g. a particular size or shape, while all other factors vary. It should be noted that the shape is the only discrete factor, but due to the continuous latent space, continuous transitions between the shapes are generated, too.

Although we observe disentanglement between the latent subspaces Z_0 and Z_1, this is not generally the case for the individual latent dimensions. While the results on the ellipsoid experiment suggest that we can, for instance, identify correspondence between latent dimensions and angular

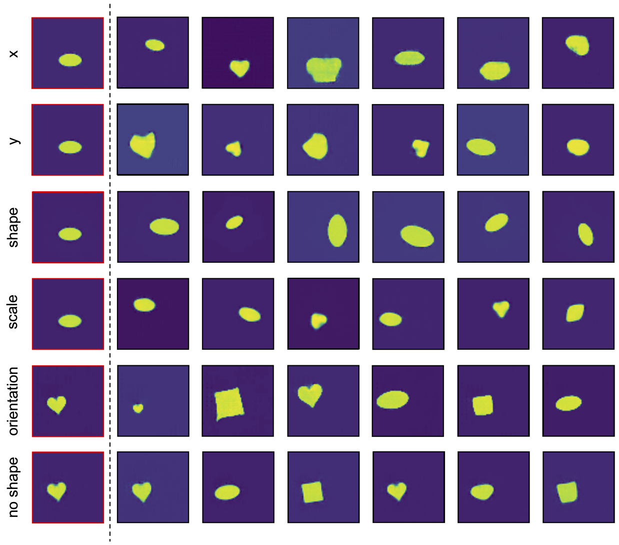


Figure 5.9: Latent traversal in invariant subspace Z_1 at a fixed property. Different sprites of the test set (reconstruction results outlined in red; first column) are selected and several models with different target properties (rows) are consider: (i) x-position, (ii) y-position, (iii) shape, (iv) scale, (v) orientation, and (vi) jointly (x, y) position, scale, and orientation (*no shape*). The coordinates in latent subspace Z_0 (i.e. property values) are fixed, coordinates in the invariant subspace Z_1 are drawn uniformly at random and the corresponding samples are generated (second to last column). The model allows fixing the target property while changing the other factors of variation, as is illustrated for the selected sprites.

components (see Figure 5.6), this is not guaranteed in general. Sampling along a particular latent dimension in subspace Z_0 or Z_1 might change several factors of variation. This is illustrated, for instance, in the latent traversal of Z_0 in one latent dimension for the *no shape* setting, where all properties vary in this latent dimension (see last row in Figure 5.8).

5.6 CONCLUSION

Our extensive evaluation corroborates that sparsity constraints and cycle consistency lead to sparse and interpretable models facilitating model selection. The results in Figures 5.5a and 5.6a demonstrate that our method identifies the sparsest solution in comparison to the standard disentanglement baseline β-VAE and the direct competitor STIB, which do not address sparsity explicitly. Furthermore, the experiments on ellipses and ellipsoids show that only our model also identifies a correct parametrisation. It correctly learns the radius r in the property subspace Z_0 as it encodes the level set, i.e. the ellipse curve or ellipsoid surface given by property Y. The angular components φ and ϑ are correctly – and in particular independently – learned in the invariant subspace Z_1 (see Figures 5.5b as well as 5.6b and 5.6c). This is a direct consequence of the cycle consistency on the property Y. It allows for semantically structuring the latent space on the basis of the semantic knowledge on property Y. Finally, these results highlight that our method is able to inherently select the correct model. Although the β-VAE and STIB are capable of attaining similar reconstruction and prediction errors, a reconstruction of level sets in these models requires a more complicated combination of latent dimensions and hinders interpretation. Therefore, only our model makes an interpretation of the learned latent representation feasible.

We showed that cycle consistency enforces conditional invariance. Tables 5.1 and 5.2 show that for all experiments, our model exhibits the best property invariance at otherwise similar reconstruction and prediction errors. The β-VAE has no mechanisms to ensure invariance and thus performs worst. But although the STIB relies on adversarial training to minimise mutual information (MI) between Z_1 and Y, the alternating training and MI estimation can pose practical obstacles, especially in cases with high-dimensional latent spaces. Our cycle-consistency-based approach has the same benefits and is more feasible. In particular, our approach can operate on arbitrarily large latent spaces in both Z_0 and Z_1, because of the inherent sparsity of the solution. Typically, an upper limit for the size of property subspace Z_0 and invariant subspace Z_1 can be defined by the dimensionality of the property Y and input X (see Figures 5.5 and 5.6). Noteworthy – although our model is trained and tested on data in the interval $[-1,1]^{d_x}$, $d_x = \{2,3\}$ – the results generalise well beyond this interval, as long as a part of the level curve or surface was encountered during training (see Figure 5.5b). This can be directly attributed to the regularisation of the latent space through additional sampling and cycle consistency of generated samples. These mechanisms impose conditional invariance which, in turn, facilitates generalisation and exploration of new samples by sharing the same level set or symmetry-conserved property.

In particular, conditional invariance can be viewed as supervised disentanglement of subspaces. Learning invariances or symmetries is closely related to disentanglement. We considered supervised settings in which semantic knowledge on a property Y serves as an inductive bias to drive the disentangling structure in the latent representation Z. This formulates the disentanglement task as performing disentanglement with a focus on a down-stream task, like predicting novel compounds with particular band gap energies, and does not require the full latent representation to factorise into disentangled latent dimensions. For representative examples of the dSprites benchmark dataset, Figures 5.8 and 5.9 qualitatively illustrate that our approach allows separating factors of variation and conditional generation. Additionally, we demonstrate the applicability of our approach in multi-property settings.

Finally, we show that conditional invariance improves targeted molecule discovery. Conditional invariance is of great importance for the generative potential of our model. In Figure 5.7 we exemplary explored the molecular structures for two reference molecules. By sampling in the invariant space Z_1, we discover molecular structures with property values which are very close to the fixed targets, i.e the mean absolute deviation is below the model prediction error. Our experiment demonstrates the ability to generate molecules with self-consistent properties which rely on the improved conditional invariance provided by our model. This facilitates the discovery of novel molecules with desired chemical properties.

6 FEATURE LEARNING AND RANDOM FEATURES IN FINITE DEEP NEURAL NETWORKS

Although deep learning achieves remarkable results in different applications in computer vision or natural language processing, the mathematical treatment of how and what these models learn is a challenging task. In that regard, the Neural Tangent Kernel (NTK) is an important milestone in the ongoing effort to build a theory for deep learning. Its prediction that sufficiently wide neural networks behave as kernel methods, or equivalently as random feature models arising from linearised networks, has been confirmed empirically for some wide architectures. We extend these findings by comparing the performance of two common finite-width convolutional neural networks, LeNet and AlexNet, to their linearisations explicitly at different network widths and on common benchmark datasets like MNIST and modified versions of it, CIFAR-10, and an ImageNet subset. We demonstrate empirically that finite-width neural networks – generally – greatly outperform the finite-width linearisation of these architectures. When increasing the problem difficulty of the classification task, we observe a larger gap which is in line with common intuition that finite-width neural networks perform feature learning which finite-width linearisations cannot. At the same time, finite-width linearisations improve dramatically with width, approaching the behaviour of the wider standard networks which in turn perform slightly better than their standard width counterparts. Therefore, it appears that feature learning for non-wide standard networks is important but becomes less significant with increasing width. We furthermore identify cases where both standard and linearised networks match in performance, in agreement with NTK theory, and cases where wide linearisations outperform their standard width counterpart. We complement the finite-width findings by considering the asymptotic limit given by the infinite-width Neural Tangent Kernel and, additionally, study the Neural Network Gaussian Process kernel.

Part of this chapter has been published in Samarin et al. (2022).

6.1 MOTIVATION

The Neural Tangent Kernel (Jacot et al., 2018) is a seminal contribution to the study of deep neural networks which extended important insights about the connection of Gaussian processes and neural networks (Neal, 1996; Williams, 1996; de G. Matthews et al., 2018; Lee et al., 2018; Garriga-Alonso et al., 2019). Ever since, subsequent investigations have refined our view on the NTK with results suggesting both its validity as well as insufficiency as an explanation for the performance of practical finite-width neural networks, and the focus of investigation has moved to the network parametrisation and architecture differences and their relationship to the NTK (Chizat et al., 2019; Lee et al., 2019; Chen et al., 2020; Hanin and Nica, 2020; Xiao et al., 2020; Seleznova and Kutyniok, 2022). The NTK framework has inspired work in many directions like infinite ensembles of trees (Kanoh and Sugiyama, 2022), federated learning (Huang et al., 2021) and thus continues to stimulate various advances in deep learning theory.

As outlined in Section 3.4.2, Jacot et al. (2018) proved that when modelling neural network training under gradient flow, i.e. full batch gradient descent of infinitesimal step size, the training trajectory $f(\boldsymbol{x}, \boldsymbol{\omega}_t)$ at iteration t satisfies an ordinary differential equation (ODE) involving the finite-width Neural Tangent Kernel

$$\begin{aligned}\Theta_t^{(L)}(\boldsymbol{x}, \boldsymbol{x}') &= \left\langle \nabla_{\boldsymbol{\omega}} f(\boldsymbol{x}, \boldsymbol{\omega}_t), \nabla_{\boldsymbol{\omega}} f(\boldsymbol{x}', \boldsymbol{\omega}_t) \right\rangle \qquad (6.1)\\ &= \sum_{i=1}^{p} \frac{\partial}{\partial \omega_i} f(\boldsymbol{x}, \boldsymbol{\omega}_t) \frac{\partial}{\partial \omega_i} f(\boldsymbol{x}', \boldsymbol{\omega}_t) \qquad (6.2)\end{aligned}$$

for weights $\boldsymbol{\omega} \in \mathbb{R}^p$ with a total of p parameters in a Multilayer Perceptron (MLP) with L layers and inputs $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^{d_x}$. The form of this kernel depends on the network architecture and the time-dependent weights $\boldsymbol{\omega}_t$ as well as a particular initialisation. In this *NTK parametrisation*, they showed that when scaling the learning rate per layer in an appropriate way and letting the width tend to infinity, the kernel converges to the infinite-width NTK $\boldsymbol{\Theta}$ which is *independent* of the weights and *stays constant* during training, greatly simplifying the ODE in this limit (see Theorem 3). Furthermore, they showed for the mean squared error (MSE) loss that the predictor at convergence is precisely what a kernel regression using the infinite-width NTK would produce (see Equation (3.75)). Importantly, the infinite-width NTK depends only on the architecture of the network; it is not learned and thus data-independent. The formalism was extended from MLPs to other architectures including convolutional networks (Arora et al., 2019; Yang, 2019b), recurrent neural networks (Alemohammad et al., 2020), residual networks (Huang et al., 2020b), transformers (Hron et al., 2020), and more general architectures (Yang, 2020).

This result can be understood as the convergence of wide networks to random feature models (Chizat et al., 2019). Let $f : \mathbb{R}^{d_x} \to \mathbb{R}^C$ be the function given by a network parametrised by weights $\boldsymbol{\omega} \in \mathbb{R}^p$ with input $\boldsymbol{x} \in \mathbb{R}^{d_x}$ and let f^i be the output in component $i \in \{1, \ldots, C\}$, with C being typically the number of classes in a classification task. For $\boldsymbol{\omega}$ sufficiently close to the random initial weights $\boldsymbol{\omega}_0$ and $\boldsymbol{u} \equiv \boldsymbol{u}_t = \boldsymbol{\omega}_t - \boldsymbol{\omega}_0$, the first order Taylor expansion in the weights

$$f^i(\boldsymbol{x}, \boldsymbol{\omega}_t) \approx f^i(\boldsymbol{x}, \boldsymbol{\omega}_0) + \nabla_{\boldsymbol{\omega}} f^i(\boldsymbol{x}, \boldsymbol{\omega}_0)\boldsymbol{u} = f^i_{\text{lin}}(\boldsymbol{x}, \boldsymbol{u}) \tag{6.3}$$

is an accurate approximation. The right-hand side $f_{\text{lin}}(\boldsymbol{x}, \boldsymbol{u})$ is a random feature model with weights $\boldsymbol{u} \in \mathbb{R}^p$ and the feature mapping $\boldsymbol{\phi}(\boldsymbol{x}) \in \mathbb{R}^{C \times p}$ is given by the gradients $\boldsymbol{\phi}^i(\boldsymbol{x}) = \nabla_{\boldsymbol{\omega}} f^i(\boldsymbol{x}, \boldsymbol{\omega}_0)$ with respect to the weights at initialisation $\boldsymbol{\omega}_0$ (see Section 3.4.2). If approximation (6.3) holds, then also the gradients $\nabla_{\boldsymbol{\omega}} f(\boldsymbol{x}, \boldsymbol{\omega}_t)$ and $\nabla_{\boldsymbol{u}} f_{\text{lin}}(\boldsymbol{x}, \boldsymbol{u})$ of the two models will be close. When training these models with some form of gradient descent and sufficiently small step size for a sufficiently small number of steps, then the training trajectories will stay close, as long as the weight vectors remain in the region around $\boldsymbol{u} = 0$ or $\boldsymbol{\omega} = \boldsymbol{\omega}_0$, respectively. Using an MSE loss in over-parametrised models, one can expect both models to converge to zero loss (Du et al., 2019). If convergence occurs before leaving this region, then the models – whether trained with early stopping or until convergence – will predict a similar function.

For the infinite-width case, Lee et al. (2019) proved that $f(\boldsymbol{x}, \boldsymbol{\omega}_t)$ and $f_{\text{lin}}(\boldsymbol{x}, \boldsymbol{u})$ converge in distribution to the same Gaussian distribution (see Theorem 4). Furthermore, the NTK result can be proved by showing that for very wide neural networks the models $f(\boldsymbol{x}, \boldsymbol{\omega}_t)$ and $f_{\text{lin}}(\boldsymbol{x}, \boldsymbol{u})$ reach zero loss and thus stop evolving before leaving the region where the approximation in Equation (6.3) is accurate (Chizat et al., 2019; Lee et al., 2019), known as *lazy training*.

The linearised model f_{lin} does not learn a representation but uses the random representation $\nabla_{\boldsymbol{\omega}} f^i(\boldsymbol{x}, \boldsymbol{\omega}_0)$ which is fixed by the initial weights $\boldsymbol{\omega}_0$ and remains unchanged throughout training. More in line with Gaussian processes and random feature models (Rahimi and Recht, 2007) but at odds with general intuition on deep learning, NTK theory predicts that, at large widths, a network and its linearisation behave similarly and no significant feature learning takes place. This seems to imply that – even for standard neural networks – *learning* a "good" representation might become decreasingly relevant with increasing over-parametrisation.

Motivated by this conjecture, we study standard convolutional neural networks (CNNs) and their respective linearisations (at initialisation) given by Equation (6.3). We complement previous work in that direction (see Section 6.2) and extend these results for more standard architectures in more common classification tasks. In particular, we perform a thorough study of two standard CNNs, LeNet (LeCun et al., 1998) and AlexNet (Krizhevsky et al., 2012), for increasingly difficult classification tasks (see Figure 6.1) at different widths.

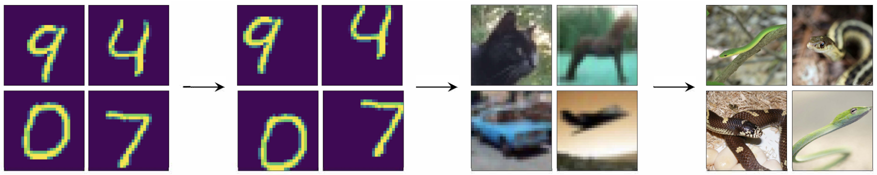


Figure 6.1: We consider four classification tasks of increasing difficulty: (i) MNIST with center-aligned digits, (ii) MNIST with random translations breaking the positional alignment, (iii) CIFAR-10, and (iv) a subset of ImageNet containing ten snake classes (see Section 6.3.1).

We observe test accuracy gaps between these networks, in line with the idea that standard neural networks perform feature learning while their linearisations do not. For the wider networks the generalisation gap closes, in line with NTK theory, supporting the picture summarised in Figure 6.2.

However, we also observe low training accuracy for the linearised networks. We investigate numerical issues which can explain reduced training performance in the linearisations and consider a simplified binary classification setting in which we can solve the linear system in Equation (6.3) with a standard solver achieving 100% training accuracy, but observe that this generally causes even worse test accuracy for the linearised models. We complement our results with a consideration of the infinite-width NTK and Neural Network Gaussian Process (NNGP) kernel, observing convergence of linearised network performance to the infinite-width NTK prediction but also standard finite-width networks outperforming this limit.

In this work, we make the following contributions:

- We show that for (nearly) all considered widths, there is a prominent performance gap between the standard and linearised LeNet and

AlexNet and this gap increases when the classification task increases in difficulty. This is shown for MNIST, CIFAR-10, and a subset of ImageNet. We believe this gap exhibits the importance of feature learning for non-wide standard networks.

- We present further instances where wide linearised networks perform as well as the standard networks and cases where linearised wide networks outperform their standard width counterpart.
- As for wider networks the generalisation gap closes, in line with NTK theory, we raise the question if this means that the non-wide and wide standard network generalise due to a very different mechanism: feature learning for non-wide networks and effectively employing unlearned random features at larger widths.
- We extend the discussion in previous work of numerical aspects of training the non-wide linearised models by considering the effective rank of the kernel.
- We show that the linearised wide networks approach the infinite-width NTK performance, but that standard wide networks can provide even better generalisation.

6.2 RELATED WORK IN CONTEXT OF THE NEURAL TANGENT KERNEL

The original motivation and prevailing appeal of (finite) deep neural networks is that they are powerful methods to extract statistics and learn features leading to strong performance for down-stream tasks (regime I in Figure 6.2) (Lee et al., 2009; Alekseev and Bobe, 2019). The behaviour of neural networks in the highly over-parametrised regime has been extensively studied, too, suggesting minor weight changes from initialisation during training (regime II) (Du et al., 2019; Allen-Zhu et al., 2019; Zou et al., 2020). In the NTK literature, typically, the infinite-width limit for finite-depth neural networks is considered (connection between regimes II and IV). In contrast, Deep Equilibrium Models consider the infinite-depth limit at finite width (Bai et al., 2019). Hanin and Nica (2020) study the NTK for both infinitely wide and deep ReLU networks, showing particular data-dependent features of the resulting NTK in these limits. Focusing on the finite-depth case, there are several studies which compare the finite-width NTK $\mathbf{\Theta}_t$ or infinite-width NTK $\mathbf{\Theta}$ to their standard network (regimes I and IV) and provide, to some extent, diverging results.

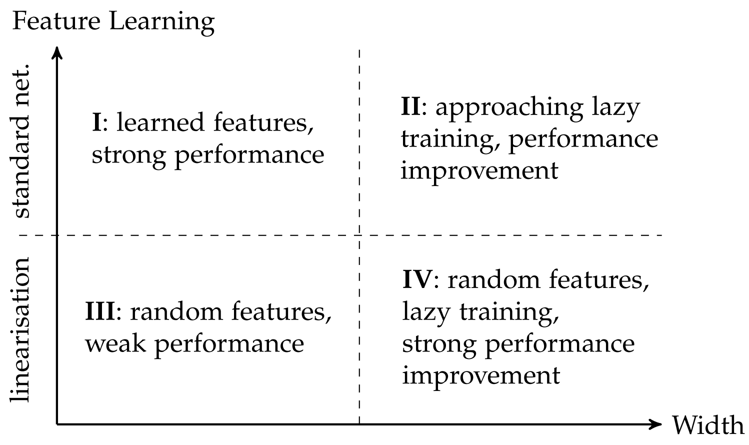

Feature Learning
standard net.
linearisation
I: learned features, strong performance
II: approaching lazy training, performance improvement
III: random features, weak performance
IV: random features, lazy training, strong performance improvement
Width

Figure 6.2: Our results on neural networks exhibit different behaviour in different regimes: For wide architectures, standard networks and their linearisation become increasingly alike. While the performance of linearised networks benefits substantially from width, standard networks only show small improvements. At usual widths, standard networks and their linearisation behave differently due to the relevance of feature learning.

The original work by Jacot et al. (2018) gives experimental results for small synthetic datasets, as well as MLPs trained on MNIST with hidden-layer-widths of 10^2, 10^3, and 10^4, showing good agreement with the infinite-width NTK for the widest network. Lee et al. (2019) extend the original work and show good agreement for small synthetic datasets and MLPs trained with SGD on MNIST and CIFAR-10. Most interestingly, a wide ResNet (Zagoruyko and Komodakis, 2016) trained on CIFAR-10 shows similar behaviour, though the non-linearised model appears to have been trained only to below 80% training accuracy, and in the test accuracy a gap seems to develop towards the end of training (see Figure 7 in their paper). In contrast to that, in Chizat et al. (2019) VGG-11 (Simonyan and Zisserman, 2015) and ResNet-18 (He et al., 2016) – trained on CIFAR-10 and widened with a scaling factor α for tuning the models into the non-linearised and linearised regimes – exhibit large gaps in test accuracy. They highlight that the decreased training performance of the linearisation is due to bad conditioning and effectively low rank of the associated kernel matrix. In their extension to CNNs, Arora et al. (2019) compare CNNs with two to 20 convolutional layers combined with fully-connected or global average pooling output layers to the derived infinite-width convolutional NTK (CNTK), observing large gaps in test accuracy on CIFAR-10.

Our work is most closely related to Lee et al. (2020) and Geiger et al. (2020). In Lee et al. (2020) an extensive empirical study of neural networks, their linearisations and the infinite-width NTK $\boldsymbol{\Theta}$ as well as the Neural Network Gaussian process (NNGP) kernel $\boldsymbol{K}_{\text{NNGP}}$ (Lee et al., 2018; de G. Matthews et al., 2018) is conducted. For fully-connected and simple convolutional architectures, they show cases where NTK can both outperform but also underperform their corresponding networks on CIFAR-10. Importantly, they study the relevance of regularisation of the kernels and identify bad conditioning of the kernel as a reason for decreased performance. In line with results by Wei et al. (2020), they show that ℓ_2-regularisation (like weight decay) of the kernel is required for good performance in practice, although this breaks the infinite-width correspondence to kernel methods. In contrast to their work, we focus on two more standard but also more extensive CNNs where we increase the widths of the standard and linearised networks explicitly and study their properties with a focus on feature learning. In that regard, our work differs from Geiger et al. (2020) who also study lazy training and feature learning but for MLPs of depth three to five and CNNs with four convolutional layers and in the framework of Chizat et al. (2019) with a scaling factor α controlling the lazy training regime. Another related line of research was conducted by Ortiz-Jiménez et al. (2021) which study linearisations with respect to task complexity defined on the basis of the NTK eigenfunctions as targets. In their evaluation on CIFAR-10, they show that linearisation performance can rank learning complexity and show that neural networks do not always outperform their kernel approximations.

Other relevant work includes Seleznova and Kutyniok (2022) which investigates the ordered and chaotic phase phenomena of vanishing and exploding gradients in the context of NTK theory, providing guarantees when the NTK is ill-conditioned (ordered phase) or well-conditioned (chaotic phase and at the border between the two phases). Furthermore, Yang and Hu (2021) note that standard and NTK parametrisations do not lead to representations that learn features in the infinite-width limit and propose an alternative parametrisation enabling feature learning in this limit.

6.3 METHODOLOGY

We examine two standard ReLU CNNs, LeNet and AlexNet, trained for classification tasks of increasing difficulty. One task is digit recognition in MNIST and modified versions which include random translations of the otherwise centered digits. In addition, we train on natural images of CIFAR-10, and a subset of ImageNet which contains ten different snake classes (see Section 6.3.1), whereby we deliberately chose similar classes to form a challenging classification task.

In this setup, we study the performance of the standard network and its linearisation f_{lin} (see Equation (6.3)) and the effect of increasing the width of the networks, thereby investigating, in particular, the relationships between regimes I and III as well as III and IV in Figure 6.2. This is done by multiplying the number of channels in each convolutional layer and all widths of fully-connected layers by a common factor, illustrated in Figure 6.3. Due to GPU memory limitations, we are able to train LeNet and *LinLeNet* up to width factors of 60 and for AlexNet and *LinAlexNet* up to width factors of 4. As the number of parameters increase quadratically in the width, and standard width LeNet and AlexNet have about 60k and 60m parameters, we were hence able to train networks of up to 212m and 912m parameters, respectively (see Appendix Section B.1 for more details).

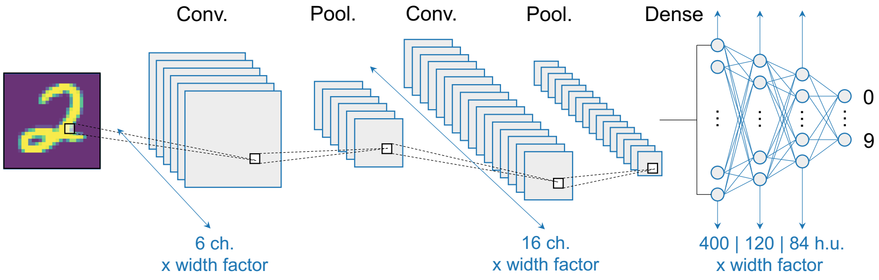


Figure 6.3: Illustration of LeNet architecture. In order to increase the width, the number of channels (*ch.*) in each convolutional layer and the number of hidden units (*h.u.*) in each fully-connected (dense) layer are multiplied with a common *width factor*.

Our implementation makes use of PyTorch's (Paszke et al., 2019) standard modules for defining and training neural networks with our own custom-made modifications for linearisation of the architectures. For LeNet, we adapt the original LeNet-5 architecture (LeCun et al., 1998) to use max pooling and ReLU activations. For AlexNet, we use the PyTorch implementation (Krizhevsky, 2014) with 10 outputs rather than 1000 (see below). Despite training for classification, we use the MSE loss with one-hot encoded target vectors. Firstly, with standard cross-entropy loss the networks never converge to exactly zero loss, so the networks must at some point leave the region where the approximation in Equation (6.3) is valid, causing some ambiguity in the heuristic. Secondly, the MSE loss allows for an easier and more efficient implementation of the training of the linearised models. Thirdly, recent results suggest that MSE loss might outperform the cross-entropy loss based on extensive evaluations on a variety of tasks in natural language processing, speech recognition and computer vision (Hui and Belkin, 2020). We furthermore do not make use of dropout, since it is not clear to us how to model it in the NTK framework (see however Novak et al. (2020)). We find that after optimising hyperparameters, we can train LeNet and AlexNet to similar train and test performance as with cross-entropy loss without dropout (see Section 6.3.1). We predict the class whose one-hot vector is closest to the output vector, which is equivalent to predicting the argmax of the output layer. We train $f_{\text{lin}}(\boldsymbol{x}, \boldsymbol{u})$ with SGD in the standard way by optimising $\boldsymbol{u}$ with gradient updates obtained by

$$\nabla_{\boldsymbol{u}} \left| f_{\text{lin}}(\boldsymbol{x}, \boldsymbol{u}) - \boldsymbol{y} \right|^2 = 2 \sum_{i=1}^{C} \nabla_{\boldsymbol{\omega}} \boldsymbol{f}^i(\boldsymbol{x}, \boldsymbol{\omega}_0) \times \left(f^i_{\text{lin}}(\boldsymbol{x}, \boldsymbol{u}) - \boldsymbol{y}^i \right). \tag{6.4}$$

Computing the gradients of the linear model with C outputs requires computing C gradients of the original network per data point, and thus C backward passes, which is computationally intensive if C is large. We therefore train (Lin)AlexNet on the snakes subset of ImageNet consisting of $C = 10$ classes, while we can use full MNIST and CIFAR-10 for (Lin)LeNet.

As our goal is to stay as close as possible to standard neural network training practices, we use SGD with weight decay and momentum, i.e. with explicit regularisation. In addition, we use the standard PyTorch weight initialisation, which is a variant of Kaiming initialisation (He et al., 2015), rather than the NTK parametrisation (see Section 3.4.2).

6.3.1 *Classification Tasks*

We consider three benchmark datasets as illustrated in Figure 6.1. Firstly, we perform recognition of handwritten digits on MNIST (LeCun et al., 1998), where a label from 0 to 9 is assigned to each individual greyscale image (28×28 pixels). Secondly, we classify the natural images (32×32 pixels; RGB) of CIFAR-10 (Krizhevsky and Hinton, 2009) which differentiates between ten different classes: airplane, car, bird, cat, deer, dog, frog, house, ship, and truck. We rescale the CIFAR-10 images to 28×28 pixels for convenience. Both datasets contain 60 000 training and 10 000 testing images. Thirdly, for a challenging classification task, we chose a subset of ImageNet 2012 (Russakovsky et al., 2015) comprised of ten snake categories illustrated in Figure 6.4. The extracted dataset contains 1300 training and 50 test images per class, resulting in 13 000 training and 500 testing images in total. As a benchmark performance result, we evaluate a standard pre-trained AlexNet on this dataset, achieving 47.6% test accuracy. Training our implementation of AlexNet with a cross-entropy loss on the snakes dataset provides 98.5% train and 51.4% test accuracy (single run). In our experiments, we used a MSE loss, which in comparison led to 99.1% train and 53.8% test accuracy (single run). These results indicate that both loss functions lead to comparable performance and outperform a standard pre-trained AlexNet (trained on full ImageNet) with respect to generalisation.

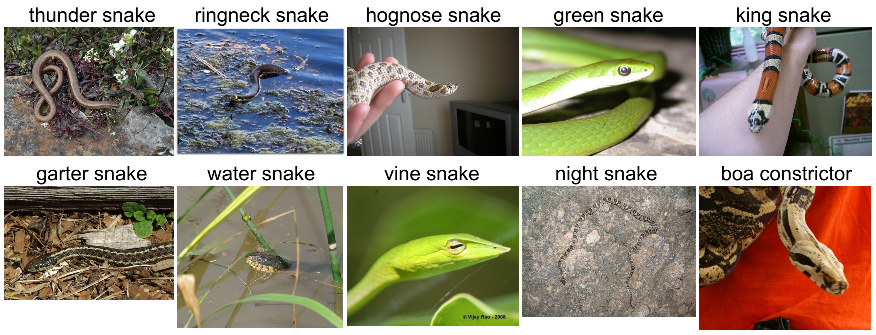


Figure 6.4: Ten snake categories from ImageNet.

6.4 EXPERIMENTAL RESULTS FOR FINITE-WIDTH NETWORKS

In the experiments, five independent reruns of the specified networks for 100 epochs and batch size 32 were performed unless stated otherwise. Hyperparameter search was conducted for each network architecture and its linearisation at all widths separately, for learning rates including $\{1, 0.1, 0.01, 0.001\}$ and weight decay values including $5 \times \{10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$. The momentum parameter was set to the default value of 0.9. For each rerun, a different fixed random seed was used to ensure that both the standard and linearised models at a particular width are initialised exactly the same and receive the same mini-batches during training. For experiments involving CIFAR-10 and the snakes dataset, the learning rate was decreased by a factor 10 every 30 epochs. Otherwise, we follow the standard preprocessing for standardising the input images and standard resizing (256 pixels) and center-cropping (224 pixels) for ImageNet images. Computations were conducted on Nvidia GeForce Titan X Pascal and Tesla V100 GPUs. For experiments involving LinAlexNet×3 and LinAlexNet×4, we used an Nvidia Quadro RTX 8000 with 48 GB memory due to the increased memory requirement. All displayed results are obtained with single precision. We also carried out all experiments in section 6.4.1 with double precision, but did not observe any striking differences.

6.4.1 *Classification with Increasing Feature Learning Requirement*

lenet trained on mnist and cifar-10: For LeNet with about 60k parameters, we used width factors ranging from 1 to 60. In all experiments involving MNIST and CIFAR-10, a learning rate of 0.1 and weight decay of 5×10^{-5} led to overall best test accuracies.

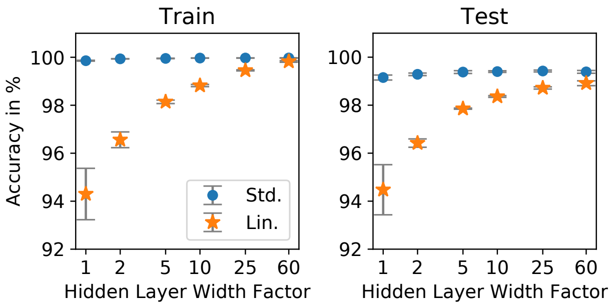


Figure 6.5: Accuracy of LeNet (•) and LinLeNet (⋆) trained on MNIST at different widths (values in Appendix Table B.3).

The results for MNIST are presented in Figure 6.5. For the standard width, a substantial difference of 4.67 percentage points in (mean) test error between LeNet and LinLeNet is observed. While LeNet does not gain appreciably from increasing the width, LinLeNet does, and the gap shrinks to 0.48 percentage points for width factor 60. Similarly, though not close in a path-wise sense, the statistics of trajectories of output values become more alike with increasing width (shown in Appendix Figure B.1), indicating a more similar behaviour of training dynamics of the linearised and standard models at large width factors.

For factor 1 the linearised model outperforms a logistic regression on normalised MNIST pixels only by a small margin, which achieves about 93% train and 92% test accuracy. The low training accuracy of the linearised models is investigated in more detail in Section 6.4.2 and 6.4.3.

When increasing the problem difficulty by randomly translating the digits horizontally and vertically, larger gaps in test (and train) accuracy are observed which also decrease with width. For factors 1 and 60, we observe 27.65 and 3.91 percentage points difference in test error. The full results are illustrated in Figure 6.6 for translations up to seven pixels (i.e. up to a quarter of the image size). We will refer to this MNIST variant as *shifted MNIST*.

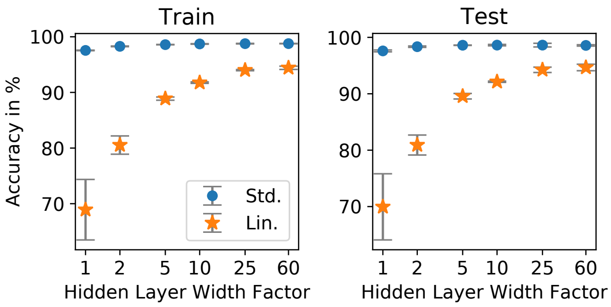


Figure 6.6: Accuracy of LeNet (•) and LinLeNet (⋆) trained on shifted MNIST at different widths (values in Appendix Table B.3). Digits were shifted randomly by up to seven pixels.

When training on the more challenging CIFAR-10 dataset even larger gaps are observed, as shown in Figure 6.7. For the standard width, a difference of 20.22 percentage points in test error between LeNet and LinLeNet is observed. This shrinks to a smaller but still appreciable gap of 13.17 percentage points at width factor 60. Interestingly, LinLeNet$\times$60 outperforms standard width LeNet$\times$1 in both training and test error (grey dashed line).

ALEXNET TRAINED ON SNAKES DATASET: For AlexNet with about 60m parameters, width factors 1, 2, 3, and 4 were used and the networks were trained on the ten-class snakes subset of ImageNet (see Section 6.3.1). For the linearised networks, a learning rate of 1 and weight decay of 5×10^{-7}

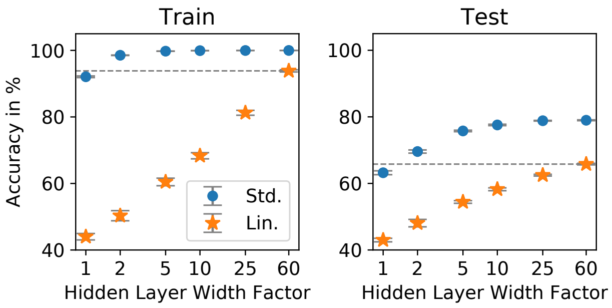


Figure 6.7: Accuracy of LeNet (•) and LinLeNet (⋆) trained on CIFAR-10 at different widths (values in Appendix Table B.3).

provided the best test errors. For the standard networks, however, a learning rate of 0.1 and weight decay of 5×10^{-6} lead to best test performance. In addition, we trained the linearised networks with these hyperparameters settings, too, for comparison. Figure 6.8 summarises the findings, which fall

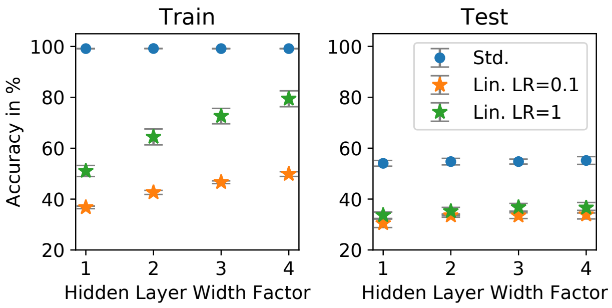


Figure 6.8: Accuracy of AlexNet (•), LinAlexNet with learning rate 0.1 (⋆) and learning rate 1 (⋆) trained on the snakes dataset at different widths (values in Appendix Table B.4).

in line with the observed trend for LeNet but give even larger gaps in test error. Trained with the same hyperparameters as their non-linearised counterparts, the gaps in test error between standard AlexNet and LinAlexNet are more than 20 percentage points at all considered widths. For the optimal hyperparameters in the linearised setting, the generalisation gap shrinks only slightly to 20.4 and 18.56 percentage points for widths 1 and 4. While increasing the width has little impact on train and test error of AlexNet, for

LinAlexNet the test error shows a slight decrease and the training error a strong decrease with width.

These results show that, at standard width or small width expansion factors, the random feature models given by the linearised networks perform poorly compared to their standard network counterpart or the random feature models of wider linearised networks. With increasing problem difficulty, the increasing gap between linearised and standard LeNet and AlexNet suggests that at standard widths significant feature learning is taking place in the standard (non-linearised) model. But with increased over-parametrisation, these gaps indeed shrink as predicted by NTK theory. The way the gap shrinks is through a dramatic improvement in performance of the linearised networks with width, while standard networks are less affected in their performance by width. However, as theory proves that the wide standard trained networks behave as random feature models, we hypothesise that the small improvements in accuracy of standard networks with width might be hiding a significant transition is the underlying *reason* for their good performance, namely from feature learning for the non-wide networks to utilising non-learned random features that apparently provide a good inductive bias for the tasks at hand for the wider networks (both linearised and non-linearised).

6.4.2 *Numerical Aspects*

The low training accuracy of the non-wide linearised models in the previous experiments raise the question of whether they are well-trained at all. Fitting the linearised model with N data points $\boldsymbol{x} \in \mathbb{R}^{d_x}$ is effectively solving the linear system

$$\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\omega}_0) = \nabla_{\omega} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\omega}_0)\boldsymbol{u} \tag{6.5}$$

for weights $\boldsymbol{u} \in \mathbb{R}^p$ and target $\boldsymbol{y} \in \mathbb{R}^C$, where p is the number of parameters of the original model and the rows of the matrix $\nabla_{\omega} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\omega}_0)$ are the gradients of each output of the network at data point $\boldsymbol{x}$. With $\boldsymbol{X} \in \mathbb{R}^{N \times d_x}$, the matrix $\nabla_{\omega} \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{\omega}_0) \in \mathbb{R}^{CN \times p}$ has $CN = 10N$ rows since one must fit each of the $C = 10$ outputs for each data point. LeNet at width factors 1 and 2 has roughly $p_1 = 60\text{k}$ and $p_2 = 240\text{k}$ parameters, respectively (see Appendix Figure B.1). Thus, the matrix $\nabla_{\omega} \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{\omega}_0)$ cannot have full rank when fitting a dataset of size $N = 60\text{k}$ (MNIST, CIFAR-10), i.e. $p_1, p_2 < 10N$, making it impossible to fit arbitrary targets. Moreover, even for wider networks it appears that matrix $\nabla_{\omega} \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{\omega}_0)$ remains effectively of low rank.

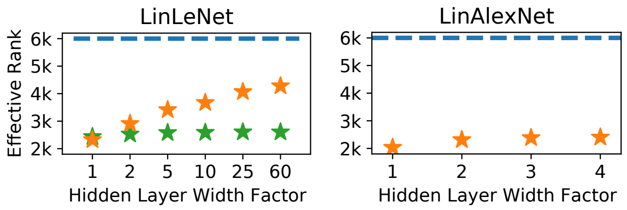


Figure 6.9: Effective rank of matrix $\nabla_\omega \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{\omega}_0)$ in LinLeNet and LinAlexNet for 600 data samples of the MNIST (standard in ⭐ and shifted in ⭐) and snakes datasets, respectively, with full rank 6000.

We quantify this by computing the *effective rank* (Roy and Vetterli, 2007) which takes the distribution of singular values into consideration and can be viewed as the exponential entropy of normalised singular values (see Appendix Section B.3). For computational reasons, we consider $N = 600$ data samples and the corresponding 6000×6000 kernel matrix $\nabla_\omega \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{\omega}_0) \nabla_\omega \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{\omega}_0)^\top$ for each width factor. For LinLeNet and considering MNIST samples, these effective ranks are much lower than the number of rows, i.e. 6000, and increase with width factor as illustrated in Figure 6.9 (left). A similar but less pronounced improvement in effective rank with width is obtained for shifted MNIST samples with additional random translation of up to seven pixels. Although AlexNet×1 with about 60m parameters (see Appendix Figure B.2) is well in the over-parametrised regime for $N = 13\text{k}$ data points and thus 130k rows in matrix $\nabla_\omega \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{\omega}_0)$, we still observe large gaps in training accuracy. As for LinLeNet, we show for 600 examples in Figure 6.9 (right) that the effective ranks at all widths are significantly lower than the number of rows of $\nabla_\omega \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{\omega}_0)$ and increase with width (marginally).

In Figure 6.10, the distribution of singular values σ of the kernel matrix is shown. We observe that increasing the width effectively increases the smallest (non-vanishing) singular values of matrix $\nabla_\omega \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{\omega}_0)$ and generally leads to a lower condition number (i.e. ratio $\sigma_{\max}/\sigma_{\min}$), for this matrix, thereby improving numerical properties.

We suspect that, in order to perfectly fit the training data, one needs to fit $\boldsymbol{u}$ also in a subspace with very small singular values, making it difficult to achieve close to 100% train accuracy with SGD with non-infinitesimal step sizes.

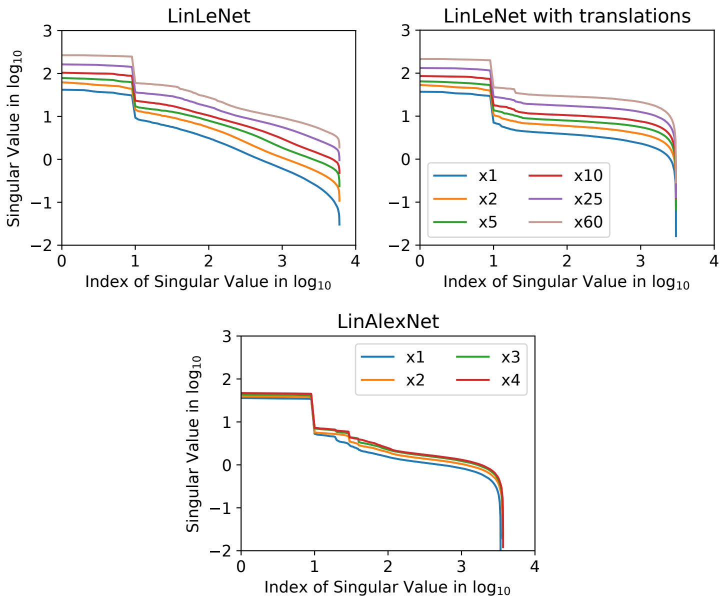


Figure 6.10: Singular value distribution of LinLeNet for 600 samples of MNIST and MNIST digits randomly shifted by up to seven pixels (upper panel) as well as LinAlexNet for 600 samples of the snakes dataset (bottom panel).

6.4.3 *Binary Classification on MNIST*

In order to study these numerical aspects in more detail, we take a closer look at the solution of the linear system in Equation (6.5). In particular, we examine if the multiclass setting might be the cause for numerical stability issues due to having multiple outputs (the different classes) for a single input, potentially leading to e.g. collinearity of rows in the matrix. Therefore, we consider binary classification with one output and train to classify a digit as 0 or not 0. Qualitatively similar results were obtained for other target classes. In the following, we solve the one-vs-rest classification task with the same least-squares objective in three ways: by training LeNet with SGD, by training LinLeNet with SGD, and by using a standard solver for linear systems. The presented results are obtained from single runs of the respective model with a fixed random seed.

SOLVING THE LINEAR SYSTEM WITH SGD: The training is performed in the same manner as before, but with a learning rate of 0.01 and for 200 epochs. Tables 6.1 and 6.2 summarise the binary classification results with target class 0 for standard MNIST and shifted MNIST. The qualitative behaviour with SGD training follows the same trend as in Figures 6.5 and 6.6 for the multiclass results (for this reason an illustration is omitted). As before, by including translations of up to seven pixels of the digits, we observe a drop in accuracies which is particularly pronounced for the linearised setting.

Table 6.1: Accuracy of LeNet, LinLeNet, and the solver on binary MNIST (0 vs. not 0) at different widths.

		×1	**×2**	**×5**	**×10**	**×25**	**×60**
Test	Solver	97.8	99.86	99.89	–	–	–
	Lin.	99.61	99.75	99.81	99.85	99.85	99.89
	LeNet	99.89	99.89	99.88	99.89	99.89	99.88
Train	Solver	100	100	100	–	–	–
	Lin.	99.42	99.74	99.88	99.97	99.9983	100
	LeNet	100	100	100	100	100	100

In comparison to the harder multiclass task, the gap in training accuracy between LeNet and LinLeNet is greatly reduced but persists for the less wide networks, especially for LinLeNet×1 in the shifted MNIST task. While training the standard network consistently leads to perfect training accuracy in the standard MNIST setting, it is not possible to achieve 100% training accuracy when solving the linear system in Equation (6.5) with SGD, in most cases. However, from a width factor of 5 on, we observe for LinLeNet in the standard MNIST task that the linearised networks start agreeing (up to the second decimal place) with the results of the corresponding LeNet. In particular, LinLeNet×60 matches the train and test results of LeNet at all considered widths, which is in agreement with NTK theory.

SOLVING THE LINEAR SYSTEM WITH A STANDARD SOLVER: Since SGD is not able to attain high train accuracy for linearised models for all widths, it raises the question whether a different algorithm can, and if so, what its generalisation properties are for the tasks at hand. An advantage of the binary classification setting is that we can directly solve the linear system

Table 6.2: Accuracy of LeNet, LinLeNet, and the solver on binary shifted MNIST (0 vs. not 0) at different widths. Input digits were randomly shifted by up to seven pixels.

		$\times$**1**	$\times$**2**	$\times$**5**	$\times$**10**	$\times$**25**	$\times$**60**
Test	Solver	86.05	98.7	99.17	–	–	–
	Lin.	95.48	98.51	98.97	99.31	99.51	99.55
	LeNet	99.72	99.80	99.82	99.77	99.82	99.86
Train	Solver	100	100	100	–	–	–
	Lin.	95.42	98.29	98.90	99.25	99.49	99.61
	LeNet	99.77	99.83	99.86	99.90	99.92	99.90

in Equation (6.5) for $\boldsymbol{u}$ for width multipliers 1, 2, and 5, as the amount of memory required to store the entire matrix $\nabla_{\boldsymbol{\omega}} f(\boldsymbol{x}, \boldsymbol{\omega}_0)$ in memory is reduced and becomes manageable. Larger widths were not feasible for us as more than 1 TB of memory is required even for binary classification, without taking additional memory requirements for the computation into account. We make use of the SciPy least-squares solver which utilises the highly optimised LAPACK library (Anderson et al., 1999). The results are included in Tables 6.1 and 6.2.

Interestingly, the solver attains perfect training accuracy in all considered cases, but at the cost of a diminished test accuracy for LinLeNet$\times$1 in standard MNIST (see Table 6.1) and, particularly, in shifted MNIST (see Table 6.2), indicating overfitting of the solver solution. Apparently, the implicit regularisation of the SGD solution significantly improves generalisation for these widths, while precluding a perfect train accuracy. For LinLeNets of larger widths, an improved generalisation is attained which we view to match the SGD results to a reasonable degree (considering fluctuations in the second decimals place as in the multiclass results, see Appendix Table B.3). In the standard MNIST task, the attained solver solutions for LinLeNet$\times$2 and LinLeNet$\times$5 match the test accuracies of their corresponding standard LeNets at otherwise 100% train accuracy. It should be noted that the solver results were obtained without regularisation. Additional regularisation should lead to similar results as for LinLeNet$\times$1 trained with SGD, that is higher generalisation and lower training accuracy.

Therefore, it appears that the observed generalisation gaps and poor performance of non-wide linearised models in Section 6.4.1 are not due to poor training optimisation. We suspect that moderately wide linearised networks in the multiclass experiments operate in a similar regime as LinLeNet$\times 1$ in the binary classification setting.

6.5 CONSIDERING THE INFINITE-WIDTH LIMIT

In the previous sections, the analysis focused on finite-width networks which nearly always exhibited a test performance gap between standard networks and their linearised counterpart.[1] In the following, we extend the discussion to the infinite-width case and study the Neural Network Gaussian Process (NNGP) kernel $\boldsymbol{K}_{\text{NNGP}}$ and the infinite-width NTK $\boldsymbol{\Theta}$ (see Section 3.4 for the theoretical background). The presented results of this section extend work done during the Master's thesis of Benjamin Kessler (Kessler, 2021).

6.5.1 *Methodology in Infinite-Width Study*

In order to study the linearisation of LeNet and AlexNet, we developed an implementation on the basis of PyTorch (see Section 6.3). Our custom implementation allows utilisation of common architecture components. Concurrently, the *neural tangents*[2] framework was introduced for the same purpose, additionally allowing computation of the kernels in the infinite-width limit. However, in order to compute these infinite-width kernels, some standard architecture components as max pooling cannot be used. In the context of the Master's thesis of Benjamin Kessler, our previous study on LeNet was reimplemented in the neural tangents framework with a few modifications.

ADJUSTMENTS IN ARCHITECTURE: As before, we adapt the original LeNet-5 architecture to use ReLU activations but replace max pooling by average pooling. Preliminary experiments on MNIST in the PyTorch implementation did not show a significant difference in performance whether using average pooling or max pooling. Furthermore, we use the NTK parametrisation (see Section 3.4.2) to initialise the weights of the networks, as opposed to standard PyTorch initialisation in the previous study. This change is required to have exact correspondence of the recursively com-

1 The only exception was LeNet$\times 60$ and LinLeNet$\times 60$ in the binary (standard) MNIST setting (see Table 6.1), where both networks match in test accuracy (at perfect training accuracy).
2 `https://github.com/google/neural-tangents`

puted kernel $\mathbf{\Theta}$ to NTK theory (see Section 3.4.2). In addition, it is possible to obtain the NNGP kernel $\boldsymbol{K}_{\text{NNGP}}$, corresponding to a *weakly-trained* infinitely wide neural network in which only the last layer is trained (see Section 3.4.1). As the finite-width counterpart to the NNGP, we consider neural networks where all weights are frozen at initialisation except the last layer and only these last layer weights are adjusted during training. We refer to this model as the *Last-Layer-LeNet* which we consider at different widths, too. The reimplemented models are referred to as *(Lin)LeNet$_{nt}$*.

ADJUSTMENTS IN THE FINITE-WIDTH TRAINING PROCEDURE: All finite-width models are trained with SGD and momentum, with the momentum parameter being set to the default value of 0.9, as before, but without explicit regulariser like weight decay. As the goal is a comparison to the infinite-width, and – in particular – infinite-long training limit, all models up to factor 25 were trained for 100 000 epochs. For computational reasons, models with width factor 60 where trained for 53 000 and 40 000 epochs on MNIST and CIFAR-10, respectively, which was sufficient for convergence. As before, we use the MSE loss with one-hot encoded target vectors. An initial learning rate of 10 was identified as an appropriate choice (Kessler, 2021) based on preliminary results of LeNet$_{\text{nt}} \times 1$, i.e. standard LeNet at standard width, but is possible that a more thorough hyperparameter search might lead to improved finite-width results. The learning rate was reduced by a factor of ten after 30 000 epochs. As before, for each model, five independent reruns with a fixed set of random seeds were performed and sample mean and sample standard deviation results are reported.

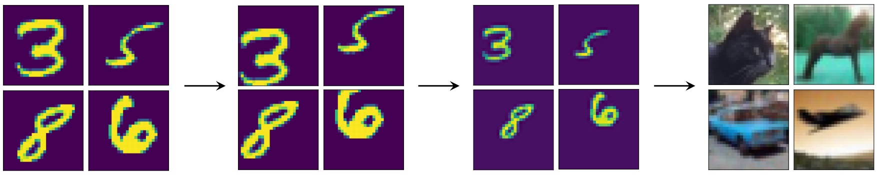


Figure 6.11: The four classification tasks considered in the infinite-width study: (i) MNIST with center-aligned digits, (ii) shifted MNIST with random translations, (iii) shifted-resized MNIST with padding, random translations, and resizing, as well as (iv) CIFAR-10.

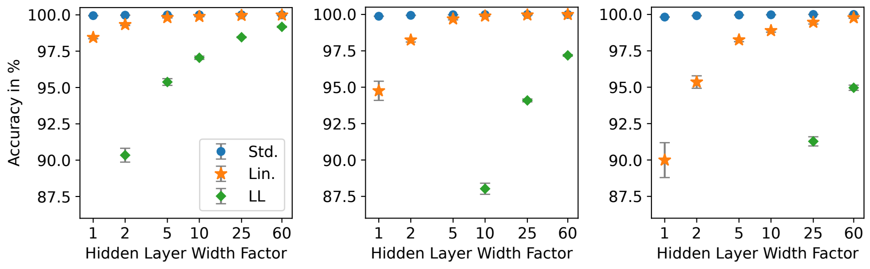
ADJUSTMENTS IN THE DATASETS: Following the general idea illustrated in Figure 6.1, we study three MNIST multiclass classification tasks. In addition to (i) standard MNIST, we consider one variant where (ii) every digit is randomly shifted by up to five pixels in horizontal and vertical direction, as well as another variant in which (iii) each input image is padded by 10 pixels at each image border, the digits are randomly shifted by up to 10 pixels horizontal / vertically and the images are resized to the original image size of 28×28 pixels. Where *shifted MNIST* only removes the positional alignment of digits, *shifted-resized MNIST* additional decreases the digit resolution and increases the difficulty to localise the digits in the images. Furthermore, CIFAR-10 in its original input image size of 32×32 pixels is considered. For computational reasons[3], the training set was restricted to (the same) 50 000 samples in all experiments. These four classification tasks of increasing difficult, i.e. from aligned digits, to shifted, to shifted and resized digits, and finally to tiny natural images, are illustrated in Figure 6.11.

6.5.2 *Revisiting Classification with Increasing Feature Learning Requirement*

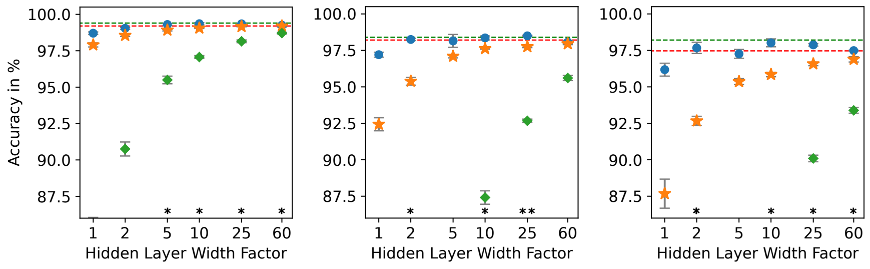
FINITE-WIDTH RESULTS: The outcomes are summarised in Figure 6.12 and Appendix Tables B.5, B.6, and B.7 for MNIST and in Figure 6.13 and Appendix Table B.8 for CIFAR-10.

For the (standard) MNIST task, the (mean) test accuracies of LeNet_{nt} exceed $\text{LinLeNet}_{\text{nt}}$ by 0.8 to 0.07 percentage points difference at widths factor 1 and 60, respectively. In the shifted MNIST task, these gaps increase from 4.78 to 0.12 percentage points difference, and in shifted-resized MNIST classification further from 8.5 to 0.6 percentage points difference at widths factor 1 and 60, respectively. In all MNIST experiments a performance gap prevails, even though LinLeNet$\times$60 on MNIST comes very closely to its standard counterpart $\text{LeNet}_{\text{nt}} \times 60$. In all settings, the test performance of wide LeNet_{nt} starts decreasing at factors of 10 or 25. This coincides with these networks attaining perfect training accuracy and might hint at an overfitting result. As expected, training only the last layer of LeNet_{nt} leads to very poor performance, overall, which however significantly improves with width. In particular, Last-Layer-LeNet$\times$60 achieves a test accuracy on par with $\text{LeNet}_{\text{nt}} \times 1$ of 98.71% and 98.70%, respectively, on MNIST. Last-Layer-LeNet$\times$60 has 50k trainable parameters similar to the 60k parameters

3 With the full training set of 60 000 samples, the computation of the NNGP kernel and infinite-width NTK was not feasible due to limited memory resources.



(a) Train accuracy.



(b) Test accuracy.

Figure 6.12: Accuracy of LeNet$_{\mathrm{nt}}$ (•), LinLeNet$_{\mathrm{nt}}$ (⋆), and Last-Layer-LeNet (⬩) trained on MNIST at different widths (numerical values are provided in Appendix Tables B.5, B.6, B.7). Dashed lines indicate the result for the infinite-width case for the NTK (red) and NNGP kernel (green) on the test set. The first column shows results for the standard MNIST dataset, the second column for shifted MNIST, and the third column for shifted-resized MNIST. LeNet$_{\mathrm{nt}}$ test accuracies which exceed the infinite-width kernel results are indicated by $*$ for the NTK and by $**$ for the NNGP kernel (see Table 6.3). Visualisations with different y-axis scaling can be found in Appendix Figures B.2 and B.3.

in LeNet$_{\mathrm{nt}}$[4]. Note that there is still a difference in the layer-wise and, in particular, non-linear architecture of LeNet$_{\mathrm{nt}}$ compared to the linear Last-Layer-LeNet model.

For CIFAR-10, a (mean) test accuracy gap of 3.84 percentage points at width factor 1 is obtained. Interestingly, the performance gap closes with increasing width and LinLeNet$_{\mathrm{nt}} \times 60$ outperforms LeNet$_{\mathrm{nt}} \times 60$ by 4.11 percentage points. However, from LeNet$_{\mathrm{nt}} \times 10$ on, the wide standard networks achieve decreasing test accuracy at perfect training accuracy of

4 See Appendix Section B.1 for an overview of trainable parameters in the considered architectures at different widths.

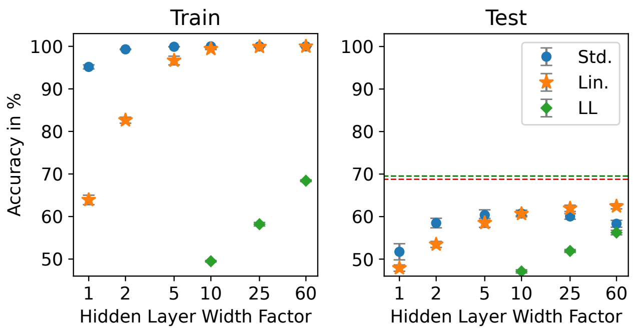


Figure 6.13: Accuracy of LeNet$_{\mathrm{nt}}$ (•), LinLeNet$_{\mathrm{nt}}$ (⋆), and Last-Layer-LeNet (◆) trained on CIFAR-10 at different widths (numerical values are provided in Appendix Table B.8). Dashed lines indicate the result for the infinite-width case for the NTK (red) and NNGP kernel (green). A visualisation with different y-axis scaling, displaying the results for Last-Layer-LeNet at all widths, can be found in Appendix Figure B.4.

100% which might be explained by overfitting (see Appendix Table B.8). In addition, it is important to note that the hyperparameter choice was based on LeNet×1 tested on MNIST and is likely not optimal for training on CIFAR-10. Therefore, these findings might not be robust to a more suitable hyperparameter choice or another optimisation approach.

Overall and in agreement with the outcomes of Section 6.4.1, the finite-width results display an increasing performance gap with increasing task difficulty. This is more the case for LinLeNet$_{\mathrm{nt}}$ than LeNet$_{\mathrm{nt}}$, but most prominently pronounced for Last-Layer-LeNet, as expected. Both LinLeNet$_{\mathrm{nt}}$ and Last-Layer-LeNet improve significantly with width, coming very close to the performance of LeNet$_{\mathrm{nt}}$. We observe again settings where a wide LinLeNet$_{\mathrm{nt}}$ performs on par or even outperforms less wider LeNet$_{\mathrm{nt}}$. This is the case, for instance, for LinLeNet$_{\mathrm{nt}} \times 25$ and LinLeNet$_{\mathrm{nt}} \times 60$ and LeNet$_{\mathrm{nt}} \times 1$ in all considered classification tasks. Furthermore, we report a wide Last-Layer-LeNet×60 competing with a standard-width LeNet on standard MNIST.

COMPARISON TO PYTORCH RESULTS: As we consider similar settings in our custom PyTorch implementation and the neural tangents reimplementation, a comparison of the results in these frameworks suggests itself. In direct comparison to the PyTorch implementation, the neural tangent variant leads to higher train accuracy but matching to (slightly) lower generalisations accuracy in all considered LeNet$_{\mathrm{nt}}$ and shared classification tasks, i.e. MNIST, shifted MNIST[5], and CIFAR-10 (see Appendix Figures B.3, B.5, B.6, B.8). In particular, the test performance of LeNet$_{\text{nt}}$ on CIFAR-10 is lower. For all LinLeNet$_{\text{nt}}$ and shared classification tasks, generally higher train and test accuracies are obtained. The only exception are the test results for CIFAR-10 of LinLeNet$\times$25 and LinLeNet$\times$60 which match or exceed LinLeNet$_{\text{nt}} \times 25$ and LinLeNet$_{\text{nt}} \times 60$.

In order to interpret these results, it should be noted that the differences in architecture (average pooling instead of max pooling), the differing NTK initialisation, different optimisation technique (no weight decay for explicit regularisation) but also the training set restriction to 50k samples complicate a fair comparison of these results. While the choice of down-sampling either through average pooling or max pooling might have less of an influence on a standard MNIST task, max pooling is favourable in most other settings, in practice.

Table 6.3: Test accuracy of the infinite-width NTK and NNGP kernel (mean prediction) for the different classification tasks.

Dataset	NNGP	NTK
MNIST	99.40	99.20
Shifted MNIST	98.40	98.21
Shifted-resized MNIST	98.21	97.47
CIFAR-10	69.53	68.77

INFINITE-WIDTH RESULTS: For these reasons, our main focus lies in the infinite-width results which – at least qualitatively – can add some additional insights into the finite-width study. The infinite-width kernels are recursively computed as outlined in Section 3.4 and the predictions are obtained by kernel ridge regression with an additional diagonal noise of 10^{-6}. The classification accuracies for the infinite-width predictions are obtained by considering the argmax in the output vector for the different classes (one-hot encoding of targets).[6] The test results are summarised in Table 6.3 and visualised in Figures 6.12 and 6.13 through dashed lines (NNGP in green, NTK in red). As for the finite-width networks, we observe

5 The two shifted settings are slightly different with up to five (neural tangents) and seven (PyTorch) pixel shifts, but the observation still applies.

6 Note that the prediction corresponds to the mean prediction and, due to the classification task and taking the argmax of the output vector, we did not take the covariance into consideration.

decreasing kernel performance with increasing task difficulty. Interestingly, we observe that the NNGP prediction slightly outperforms the NTK in all cases, with similar results being reported by Lee et al. (2020)[7]. The relative performance difference increases with increasing task difficulty. This is a surprising results due to the correspondence of the NTK to a *fully-trained* network, while the NNGP can be viewed to correspond to a *weakly-trained* network. In agreement with NTK theory, the LinLeNet$_{\text{nt}}$ mean performance approaches (from below) the infinite-width NTK mean prediction with increasing width. In particular, LinLeNet$_{\text{nt}}$ $\times$ 60 with a mean test accuracy of 99.19% is close to the NTK result of 99.20% on MNIST. However, some standard LeNet$_{\text{nt}}$ exceed the infinite-width NTK prediction and in one case even the NNGP prediction, as highlighted in Figure 6.12b. Still, for larger widths, these predictions converge (from above) to the infinite-width NTK prediction. In particular, LeNet$_{\text{nt}}$ $\times$ 60 with a test accuracy of 99.26% comes close to the NTK and LeNet$_{\text{nt}}$ $\times$ 60 results of 99.20% and 99.19% on MNIST. On the one hand, these results highlight that linearisations indeed converge to the NTK prediction, at least in the extent of our experiments. On the other hand, benefits of neural networks of finite width are highlighted, which seem to imply that modes of generalisation can be identified which exceed the generalisation modes attained in the infinite-width limit. However, it is proved that standard networks and their linearisation (in the consider NTK parametrisation) converge to the same performance in the NTK limit (see Theorem 4). Consequently, this seems to imply that there is a *critical* width or over-parametrisation from which generalisation capability of standard networks deteriorates, like factors 10 or 25 in the considered experiments. An important open question, however, remains the role of regularisation in these results. Although regularisation (implicit or explicit) was employed in both the neural network training and kernel regression, the impact of varying the regularisation strength is potentially an interesting future research direction.

6.5.3 *Numerical Considerations with Random Labels*

We perform an additional MNIST experiment in which the relation between input image and label is removed. To this end, we randomly generate a new label between 0 and 9 for each input image. This approach is inspired by Zhang et al. (2017, 2021) which showed that common over-parametrised neural networks – in many cases – can achieve (almost) perfect training

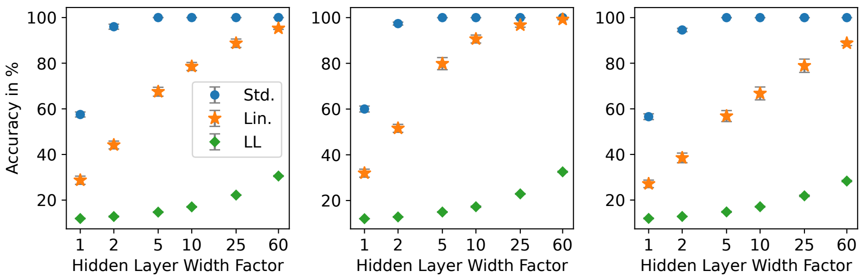
7 It should be noted that both studies use the neural tangents framework.

accuracy on even randomised labels. Importantly, this is possible to achieve without adjustments to the learning problem, in particular no hyperparameter changes are required, and training time is only increased by a small constant factor in the random label setting compared to using the correct labels. In other words, over-parametrised neural networks *easily memorise* even noise, but of course do not generalise to the test set. In Figure 6.14 we visualises the results for LeNet$_{\mathrm{nt}}$, LinLeNet$_{\mathrm{nt}}$, and Last-Layer-LeNet trained on MNIST, shifted MNIST, and shifted-resized MNIST with the same random labels and the same hyperparamters as previously. The sample mean accuracy as well as sample standard deviation for five independent reruns of models are reported.

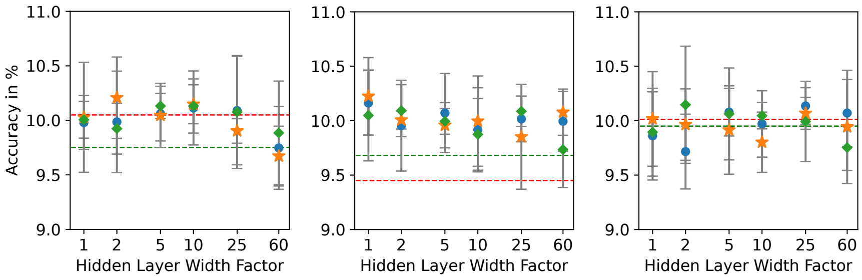
As noted earlier, considering LeNet on the MNIST classification tasks, the network and its linearisation are over-parametrised from width factor 5 on (see Appendix Figure B.1). In line with the expectation, over-parametrised LeNet is capable of attaining perfect training accuracy but does not generalise in the test setting, with a generalisation error of about 90% in all cases which corresponds to random guessing on the test set. This expected generalisation behaviour is also shown for LinLeNet$_{\mathrm{nt}}$, Last-Layer-LeNet, and the infinite-width NNGP and NTK results. However, over-parametrised LinLeNet$_{\mathrm{nt}}$ does not fit training examples perfectly like its standard counterpart, and shows similar behaviour of increasing accuracy with width as previously seen. These results corroborate our findings in Sections 6.4.2 and 6.4.3 of numerical aspects precluding perfect fitting of training data with SGD in the linearised setting, i.e. numerical properties of the linear system in Equation (6.5) and matrix $\nabla_\omega f(X, \omega_0)$ as well as the role of (implicit) regularisation. Interestingly, LinLeNet$_{\mathrm{nt}}$ generally achieves higher train accuracies in shifted MNIST than standard MNIST which again might hint at improved numerical properties of the matrix $\nabla_\omega f(X, \omega_0)$, e.g. reducing collinearity of rows in the matrix. It should be noted that no modifications to the hyperparameters were considered and that it is likely that tuning hyperparamters for training LinLeNet$_{\mathrm{nt}}$ can lead to generally higher training accuracies.

6.6 CONCLUSION

Motivated by conflicting results in NTK literature, we studied two classical convolutional neural networks, LeNet and AlexNet, and their corresponding linearisations at different widths and increasing difficulty of classification tasks in two different frameworks. We investigated four regimes of different



(a) Train accuracy.



(b) Test accuracy.

Figure 6.14: Accuracy of LeNet$_{\text{nt}}$ (●), LinLeNet$_{\text{nt}}$ (⋆), and Last-Layer-LeNet (◆) trained on MNIST with random labels at different widths (values in Appendix Tables B.9, B.10, B.11). Dashed lines indicate the result for the infinite-width case for the NTK (red) and NNGP kernel (green). The first column shows results for the standard MNIST dataset, the second column for shifted MNIST, and the third column for shifted-resized MNIST. In all cases the same randomised labels were used.

behaviour in neural networks (see Figure 6.2) which complement previous results on lazy training (Chizat et al., 2019) and random feature models (Lee et al., 2019, 2020) summarised in the following.

Firstly, in agreement with previous results like by Arora et al. (2019), we observed significant train and test performance gaps between standard width LeNet and AlexNet and their corresponding linearisation. By considering different classification tasks of increasing difficulty, we showed that the performance gaps increase accordingly suggesting that richer features need to be learned, which the effectively random feature models LinLeNet and LinAlexNet cannot provide.

Secondly, in agreement with work such as Jacot et al. (2018); Lee et al. (2019), we showed, however, that width improves the performance of linearised networks significantly. We hypothesise that the comparatively minor improvements in performance of standard networks might hide a transition from feature learning to utilising random features at moderate widths. This might be related to previous results suggesting that the intermediate representations of networks of increasing width become increasingly alike to each other and to the representation in the large width limit (Kornblith et al., 2019).

Thirdly, we showed that numerical aspects like the effective rank (see Figure 6.9) and distribution of singular values (see Figure 6.10) of the feature mapping $\nabla_\omega \boldsymbol{f}(\boldsymbol{X}, \boldsymbol{\omega}_0)$ have a role in explaining low training accuracy of SGD trained non-wide linearised models. Increasing width appears to remedy these numerical issues of the associated kernel.

Lastly, in the additional infinite-width investigation, we showed that the linearised models converge to the infinite-width NTK result but wide standard networks can achieve better generalisation results. This might imply that there is *critical* width or over-parametrisation from which generalisation capability of standard networks deteriorates. But also the relevance of regularisation in the neural network training and kernel ridge regression appears to be a relevant direction of future research.

In summary, our first investigation is based on the finite-width NTK at initialisation and explores deviations, as described above, but also convergence to NTK theory. In particular, we show agreement in performance of standard networks and their linearisation as well as an instance where a wide LinLeNet($\times 60$) outperforms its standard width LeNet($\times 1$) on CIFAR-10. Our second investigation adds the infinite-width perspective. We provide further results where wide LinLeNets and LeNets match in performance, where wide LinLeNets outperform their standard-width LeNet counterpart but also cases where wide LeNets outperform the predicted infinite-width limit of the NTK and, in one case, NNGP.

Our study highlights the need to study theoretical descriptions of neural network generalisation beyond the finite-width NTK at initialisation, for instance by considering time-dependent NTK (Huang and Yau, 2019; Jacot et al., 2018) for finite-width networks (see e.g. Fort et al. (2020)) or by further developing the various proposed *mean-field* theories (Chizat and Bach, 2018; Hu et al., 2019; Javanmard et al., 2019; Mei et al., 2019; Nguyen, 2019; Rotskoff et al., 2019). Additionally, it highlights the need to study the nature of the potential transition to effectively random features at moderate widths in standard neural network training.

7 CONCLUSION

Devising task-relevant representations is at the core of many machine learning approaches. Learning appropriate representations from data without a necessity for feature engineering is a key reason for the success of deep learning. Even though conceptually the general objective of representation learning might be obvious, i.e. transforming an input space to a feature space to address a task with small error, it is not immediately clear how to formulate the objective to guarantee *good* representations and how these are selected by a neural network. For this reason the topic of this thesis is to identify informed ways of representation learning relevant to challenging real-world tasks and study representations more broadly. In this final chapter, we summarise our main findings and add a discussion of the individual contributions with respect to the overarching topic of informed representation learning as well as a consideration of limitations and future directions.

7.1 SUMMARY

Reflecting on the research questions formulated in section 1.2, we provide the following answers and contributions.

Firstly, we developed an approach for efficient and scalable semantic segmentation of degraded soil in Swiss alpine grasslands based on the U-Net. As we have shown in Chapter 4, we are able to identify areas affected by shallow landslides, livestock trails, sheet erosion, and (land-use) management in agreement with results of the more established object-based image analysis (OBIA) and, in particular, obtain matching linear trends of increasing total degraded area in the Urseren valley between 2000 and 2016. While OBIA is well-suited for small scale applications, like individual valleys, our U-Net approach enables mapping of erosion sites on alpine-wide scale. Although the U-Net approach was able to retrieve previously OBIA-mapped erosion sites to a high degree (intermediate to high recall scores), we observe over-segmentation in the U-Net results contributing to more segments being detected as relevant erosion sites which are not mapped in OBIA (and accordingly intermediate to low precision scores).

These deviations from the OBIA baseline highlight a particular challenge in this application: Based on remotely-sensed imagery (RGB spectrum) as in our study, a definite assignment of whether a site is belonging to one of the erosion classes is challenging in many cases, even for domain experts. The additional U-Net segments can identify valid erosion sites potentially missed in the OBIA baseline. Therefore, assuming an exhaustive baseline segmentation is difficult and, to some extent, ambiguity in the ground truth needs to be expected in this application.

Secondly, we address the question of what makes a *good* representation posed in the beginning of this chapter and consider learning latent encodings which allow preserving symmetries in a supervised problem setting. In Chapter 5, we introduce a generative model – based on the deep variational information bottleneck (DVIB) – which allows separating property information from other input object information. Our method can be viewed as a supervised disentanglement approach which partitions the latent space into a disentangled property-relevant subspace and a property-invariant subspace. By ensuring that generated objects fulfil cycle consistency on the (continuous) property, i.e. generated objects processed once more by the encoder and property decoder lead to similar property values, our method enforces the conditional invariance necessary for the subspace disentanglement. We demonstrate this disentanglement capability with respect to individual and multiple properties on a benchmark dataset. Our method outperforms state-of-the-art methods in property invariance. Thereby, we improve targeted generation of novel objects like compounds with desirable physical or chemical properties. Due to the regularising effect of cycle consistency through the semantic knowledge on the property and a more general sparsity constraint on the encoder, our method provides mechanisms for built-in model selection and improves interpretability as well as exploration of the latent representation.

Thirdly, we investigate feature learning and random feature models in the context of the Neural Tangent Kernel (NTK). In Chapter 6, we empirically compare two standard convolutional neural networks (CNNs) of increasing width with the random feature models given by linearisation of these networks at initialisation. We additionally study randomly initialised CNNs of different widths where only the last layer is trained and consider the Neural Network Gaussian Process (NNGP) kernel and the NTK for weakly and fully-trained infinitely wide CNNs, respectively. We show that prominent performance gaps between finite-width standard and linearised networks are observed which increase with the difficulty of the classification task and generally decrease with increasing width of the CNNs. In a few cases, we show matching results between wide standard and linearised networks in correspondence to NTK theory, and also cases where wide linearised networks, i.e. random feature models, exceed standard networks in generalisation performance. Comparing our findings suggests that there might be a transition of the considered standard CNNs from feature learning to effectively employing random features at moderate finite widths, as random feature models substantially improve with width and approach the performance of wide standard networks. We further observe that CNNs of finite-width can exceed the generalisation performance obtained in the infinite-width limit. This further suggests a critical limit for over-parametrisation in standard CNNs, from which having more parameters might hurt the generalisation capability. We discuss our results in context of numerical aspects like numerical challenges in solving the linear systems of equations with SGD and the role of regularisation.

7.2 ON INFORMED REPRESENTATION LEARNING

The numerous U-Net extensions covered in Section 3.6 attempt to improve the learned representations in several informed ways. Be it by refining the multi-resolution skip connectivity to more dense structures connecting different frequency regimes, learning to attend to different parts of feature maps connecting more relevant signals, or lending ideas of transformer architectures to incorporate long range correlations which exceed the receptive field of convolutional filters. In the context of potentially ambiguous target objects and ground truth segmentations, probabilistic approaches to semantic segmentation like the Bayesian U-Net (Dechesne et al., 2021) or the (hierarchical) probabilistic U-Net (Kohl et al., 2018, 2019) are particularly relevant. The latter approach allows the possibility of several hypotheses for the segmentation result of an input image to be included in the model formulation. We demonstrate the relevance for medical image segmentation in Figure 7.1. In the illustrated example, four different experts outlined a potential lung lesion in a computed tomography (CT) scan leading to three different ground truth segments and one instance where the expert did not recognise any lesion. Due to diffuse boundaries and limited precision in the ability to outline lesions, but also due to different expert assessment, an objectively correct segmentation is likely not achievable in all cases. In many other cases, the outlines of the experts differ only slightly, while in some cases experts disagree more on the outline or whether any lesion is

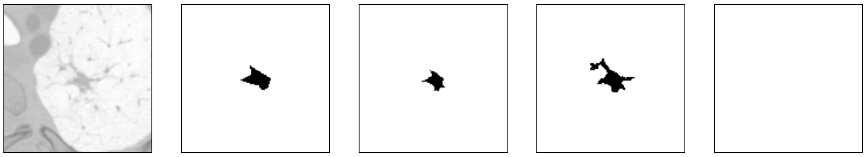


Figure 7.1: Lung CT scan example (first image) with a potential lesion manual outlined by four different experts (second to fifth image) taken from the LIDC-IDRI dataset (Armato et al., 2011). Three experts identified a lesion of varying shape, while the last expert did not recognise any lesion in the CT scan.

present. We argue that the erosion segmentation task can be viewed to be of a similar kind. In Figure 7.2 we show an area in the Urseren valley affected by livestock trails with the corresponding OBIA mappings for aerial images of 2010, 2013, and 2016. In these examples, the majority of livestock trails persist in the considered time span. However, the coverage of mapped trails varies for the different years and some segments identified in one year are missing in other years, although similarly (visually) present in the aerial images. The underlying reason is that the OBIA segmentation needs to be obtained for each aerial image individually, leading to potentially systematic differences. This is not only the case for aerial images of different years for the same valley, but in particular for different valleys, too. Similar to the medical example, the segmentation result for a individual aerial image may be viewed as the assessment of a separate expert, i.e. the OBIA settings in this particular case, with different experts leading to differences in the segmentation result. This would even be the case for human experts assessing the presence and extent of erosion sites. The discrepancy can be less divergent in some cases, but more prominent in others. Variability in the ground truth segmentation can be one of the reasons for the observed over-segmentation in the U-Net results, which attempts to learn from these partially disagreeing training instances. A modelling approach to incorporate this ambiguity of ground truth segments was performed in the Master's thesis of Manvi Bhatia (Bhatia, 2020). Based on the probabilistic U-Net (Kohl et al., 2018), a combined U-Net and VAE architecture was trained where samples of the latent space enable multiple segmentation hypotheses. Our results showed preliminary improvements in segmentation results as compared to a standard U-Net approach like in Chapter 4. Devising the model to explicitly reflect potential ambiguity in the target segmentation enables more informed representation learning.

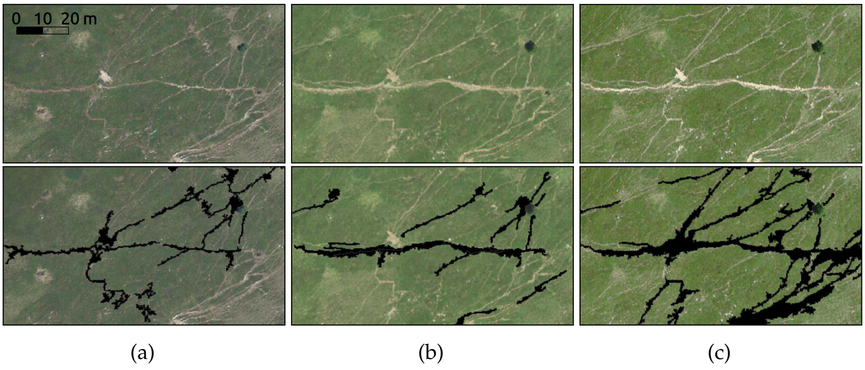


Figure 7.2: Examples of an area in the Urseren valley affected by livestock trails. The upper panel shows the aerial images for (a) 2010, (b) 2013, and (c) 2016 with the corresponding OBIA mappings of livestock trails in the lower panel.

A potential source for the ambiguity in the underlying baseline segmentation can be different light conditions (due to e.g. sun position and cloud cover) or seasonal factors which lead to a different visual appearance to which the OBIA pipeline is particularly susceptible to. Our approach on disentangling the latent representation introduced in Chapter 5 specifically targets settings where semantic knowledge on properties or covariates can be used to encode conditional invariance. This allows more informed representation learning which we showed on a molecule and disentanglement dataset, but could also be used to improve segmentation of e.g. erosion phenomena by considering relevant covariates like precipitation, cloud cover and others. Possible extensions are discussed in the context of limitations and future directions (see Section 7.3).

In the more general consideration of representation learning in Chapter 6, we highlighted benefits of finite-width convolutional neural networks. We empirically showed that wide CNNs perform feature learning leading to generalisation performance which appears to exceed the infinite-width generalisation performance of these CNNs, as considered in the context of NTK theory. Our results seem to imply that increasing over-parametrisation does not necessarily improve generalisation. This is at odds with the common setting for state-of-the-art neural network architectures and observations that over-parametrisation leads to improved generalisation and the double descent phenomenon (see Section 3.3). Although we highlight the empirical and speculative nature of our result, this finding would mean an extension to the double descent phenomenon: At a certain degree of over-parametrisation, the generalisation error might increase again, leading to a global minimum (see Figure 3.3).

With increasing width, the generalisation performance of standard networks and their corresponding random feature models appear to approach each other, in line with results on over-parametrised neural networks converging to Gaussian processes. In a similar vein, previous work has shown similar findings for particular kinds of random features, like random Fourier features or features of randomly shifted grids with random resolutions (Rahimi and Recht, 2007), or also general random features models (Rahimi and Recht, 2008) similar to the Last-Layer-LeNet considered in our work, which require more non-linearities (i.e. width in our setting) for matching performance in accordance to what we have reported in Figures 6.12 and 6.13. Furthermore, our results are in agreement with other approaches using random basis functions or feature mappings followed by a linear model (corresponding to our Last-Layer-LeNet) like Extreme Learning Machines (Huang et al., 2006) or the more classical radial basis function (RBF) networks (Broomhead and Lowe, 1988; Park and Sandberg, 1991). Therefore, our work complements previous research on random feature models but is motivated from NTK theory, and also highlights benefits of representation learning in wide neural networks for generalisation.

7.3 LIMITATIONS AND FUTURE DIRECTIONS

In the previous section, we already alluded to certain limitations and potential extensions of our work. In the following, we summarise persisting challenges and provide potential future directions.

7.3.1 *Assessment of Soil Degradation with Deep Learning*

Soil degradation detection in alpine grasslands is a challenging task for several reasons. Direct inspection of susceptible areas is infeasible due to difficult access to these terrains and the mere extent of relevant areas, allowing only selective site visitations. Thus, tackling this task needs to rely on remotely sensed information like satellite or aerial images. However, these typically come at the cost of either low temporal or spatial resolution. Due to the outstanding geodata provided by swisstopo, we had access to high-resolution aerial images for Switzerland which, however, are usually recorded only at larger time steps of about three years. Although some of our considered erosion processes, like shallow landslides, lead to longer lasting soil degradation than this period, other classes like management effects are expected to be more transient and show more seasonal variability. Even though OBIA is a suitable and highly accurate method for mapping such aerial images, the consistency of segmentation results can vary, as illustrated in Figure 7.2. In particular, the stability of results might be impacted by short-scale events like heavy rainfall or droughts and can be influenced by particular light conditions – due to e.g. varying sun position and differing cloud cover conditions – leading to a different visual appearance. Because of the low temporal resolution, the baseline segmentation is disproportionately affected.

As we have shown, the U-Net approach allows for a reliable, more automated and (possibly) more objective segmentation of degraded soil, as it attempts to retrieve the relevant statistics and expert knowledge put into the OBIA segmentation. However, because the learning approach relies on these baseline results, the instability is reflected in the U-Net segmentation results, too. In particular when training with many valleys and thus many independent OBIA settings, it is assumed that this leads to the observed over-segmentation and some more prominent jumps in degraded area results from one year to another (see Figures 4.15 and 4.16). Individual segmentation results and findings on temporal and spatial trends have to be considered under this source of uncertainty. It appears unlikely that, even with an *optimal* OBIA workflow, this instability can be fully addressed. Therefore, an important current limitation of the proposed semantic segmentation approach for erosion phenomena has to do with the availability and stability of the ground truth segmentation.

There are two future directions worthwhile pursuing. Firstly, as a natural characteristic of deep learning approaches, investing in more extensive and different data sources can address the instability. The U-Net approach was specifically chosen due to its applicability in domains with small training sets. Making use of other remote sensing sources, like satellite or unmanned aerial vehicles (UAVs)[1] imagery, can increase the temporal resolution by providing additional images at smaller timescales. This might capture temporal developments more appropriately and alleviate the effect of short-scale events. Other or broader spectra than the used visible spectrum (RGB), like near-infrared, can provide additional information e.g. on vegetation. Secondly, on the model side various avenues are possible. Concurrently to our project, several U-Net extensions were proposed (see Section 3.6) of which the U-Net variants dealing with probabilistic approaches and

1 Commonly referred to as *drones*.

ambiguity appear particularly expedient. In that regard, a combination of our first two contributions can be considered, i.e. an extension to probabilistic segmentation with side information. In addition to the aerial imagery and surface properties like slope, aspect, and curvature as used in our study, further predictors like precipitation, land cover, snow days, water accumulation, and geology, to name a few, are available. However, these predictors can come at highly different resolutions and usually aggregate information on grids of lower resolution than the aerial images. In many cases, this kind of side information can be viewed as an additional label for an image patch. Extending a U-Net-like architecture for segmentation and label prediction with a latent space encoding as proposed in Chapter 5 enables conditional generation of segmentation hypotheses depending on particular labels. These labels can be, for example, a proxy for the sun position or yearly precipitation rates, which have an influence on the light conditions or general susceptibility for soil erosion to take place, respectively. Preliminary experiments with such a model yield promising results for working with ambiguity in semantic segmentation tasks, but are left for future work.

As a different model extension, the maps on predictors such as precipitation and others can serve as additional input layers in U-Net approaches making use of the attention mechanism or transformer-like models to assist in identifying different input modalities relevant for successful segmentation of erosion phenomena.

7.3.2 *Structuring Latent Representations with Conditional Invariance*

A common take on feature learning is to consider representations as appropriate if they disentangle as many underlying factors of variation in a dataset as possible, while discarding as little information as necessary (Bengio et al., 2013). In our model, we use a training criterium based on cycle consistency of property prediction which allows disentanglement of property-relevant and property-invariant latent variables limited to a supervised setting. Our application aims at targeted object generation by including semantic knowledge on relevant target properties, and as such we were only interested in quantifying property invariance (see Tables 5.1, 5.2, and 5.3). General disentanglement approaches usually consider the unsupervised setting and quantify latent disentanglement with a variety of different measures (Locatello et al., 2019). Future work could extend our model with elements of unsupervised disentanglement approaches to improve disentanglement in the latent subspaces and study properties as well as disentanglement metrics of our conditional invariance approach.

Another view on our contribution is manifold learning. We introduce our approach from the perspective of learning mappings which preserve symmetries. Our goal can be formulated as characterising manifolds which are implicitly defined by level sets of property values in the input space. In our study, we mostly focus on simple cases of connected and convex level sets and assume that there exists a global parametrisation of level sets in the considered applications. In a follow-up extension, we introduce a novel class of flexible *generalised input-convex neural networks*. By design, these models are guaranteed to learn connected level sets which are globally parametrisable and form smooth manifolds in the input space (Nesterov et al., 2022). Further extension might investigate representing disconnected level sets corresponding to a partitioning of the input space.

7.3.3 *Representation Learning in Over-Parametrised Neural Networks*

The theory of deep learning does not keep up with the pace at which novel neural network approaches are proposed. Advancements in this direction, however, are essential as some findings on the performance of neural networks seem not to be compatible with classical statistical learning theory, as for example the double descent phenomenon. In our work, we set out to study standard convolutional neural networks from the perspective offered by NTK theory and shed light on contradictory results in this context. Although placed in a more theoretical setting, our work is limited to an empirical investigation of (in most cases) over-parametrised neural networks and thus can only provide pointers for future directions. Another important limitation in this line of work is the resource intensive nature of training (and tuning) increasingly larger networks of several hundred million to almost a billion parameters. While our linearised models, in theory, are substantially easier to train than their standard counterparts with non-linear compositions of several layers, we did not observe a speed-up in training of linearised models compared to standard models. In general, linearised networks of the same width required a similar if not slightly larger training time. In our own custom PyTorch implementation, this is mostly due to the fact that the random feature mapping had to be computed every time anew, as we could not store the matrix in GPU memory, and several backpropagation steps had to be computed with each parameter update. Additionally, in the neural tangents implementation no speed-up was observed, either, with similar training times in both linearised and standard training.

In future work, a more detailed consideration of numerical aspects might be required in order to refine the findings of this thesis. For instance, an extended investigation on the role of regularisation of the kernel and network training appears relevant. With this, more conclusive statements about generalisation performance with increasing width and the infinite-width limit might be feasible. In the latter case, including the covariance for quantifying the uncertainty in the classification task should be considered.

A very interesting research opportunity presents itself in studying the conjectured transition of a feature learning regime to effectively employing random features in standard neural networks at critical levels of over-parametrisation. A possible direction in that regard could be to study the similarity of feature representations at different widths or over-parametrisation levels, employing metrics like centered kernel alignment (CKA) or other measures of similarity or randomness of feature representations (Kornblith et al., 2019; Jones et al., 2022).

Understanding the interpolating regime of highly over-parametrised neural networks, which appear to show intimate connections to random feature models and Gaussian processes, might facilitate developing a theory on representation learning and the surprising generalisation capabilities of deep neural networks.

7.4 CLOSING REMARK

This thesis focused on improving and understanding aspects of algorithmic modelling in deep learning to address challenging tasks. Concurrently to this *model-centric* approach, *data-centric* machine learning shifts the focus to the quality of datasets. While devising novel ways of incorporating inductive biases into the model formulation offers new opportunities, putting an effort into curating extensive, high-quality datasets also has a strong impact on model performance and facilitates representation learning (Sun et al., 2017). In particular for the deployment in practical real-world applications, this reliable data basis is indispensable. Despite the remaining challenges, machine learning can provide highly useful tools for monitoring soil degradation and developing novel compounds or drugs, as shown in this thesis. In a great variety of other applications, machine learning also has the potential to assist professionals who are faced with an abundance of information to make sense of and thus improve our ability to draw conclusions.

A INFORMED LATENT SPACE ENCODING THROUGH SIDE INFORMATION

A.1 LATENT TRAVERSAL RESULTS

In the following, we examine what different latent dimensions in our model encode for. To this end, we consider – in both the ellipse and ellipsoid experiment – ten equidistant values in the selected Z_0 dimension 1 (see Figures 5.5a and 5.6a) and sample points in the remaining Z_1 dimensions 4 to 8 by varying coordinates in one of these latent dimensions while keeping all other latent dimensions fixed. Figures A.1 and A.2 illustrate the latent traversal results for the ellipse and ellipsoid experiments, respectively. The different colours represent fixed values in Z_0. We observe that for the ellipse setting only latent dimension 8 and for ellipsoid setting only latent dimensions 6 and 8 encode relevant information. Sampling solely in these selected latent dimensions reconstructs the full ellipses and ellipsoids.

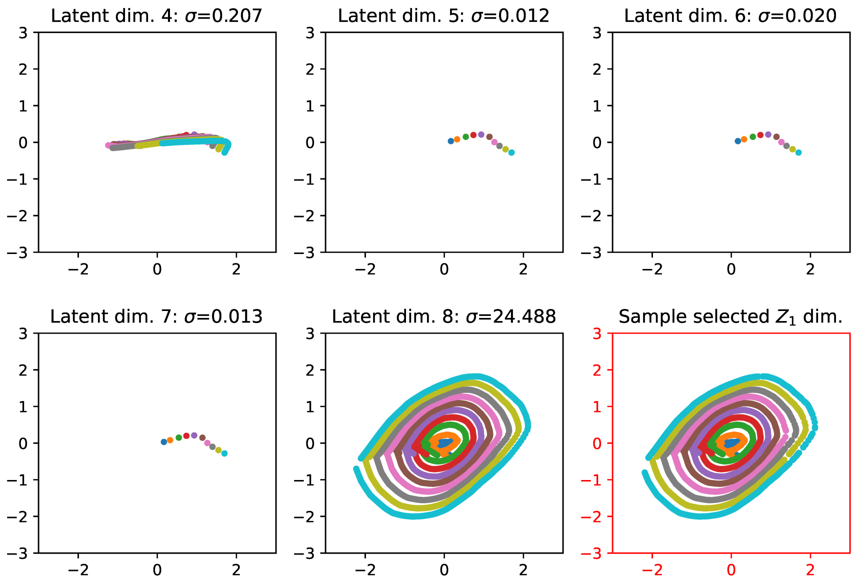


Figure A.1: Illustration of latent traversal in our model for the ellipse experiment and latent dimensions $4-8$ in the original input space ($d_x = 2$) for fixed values in the property space dimension 1 (different colours). The selected dimension 8 represents the angular component φ and reconstructs the full ellipse curves. The last plot (red borders) samples in all selected dimensions, which in this case is only dimension 8.

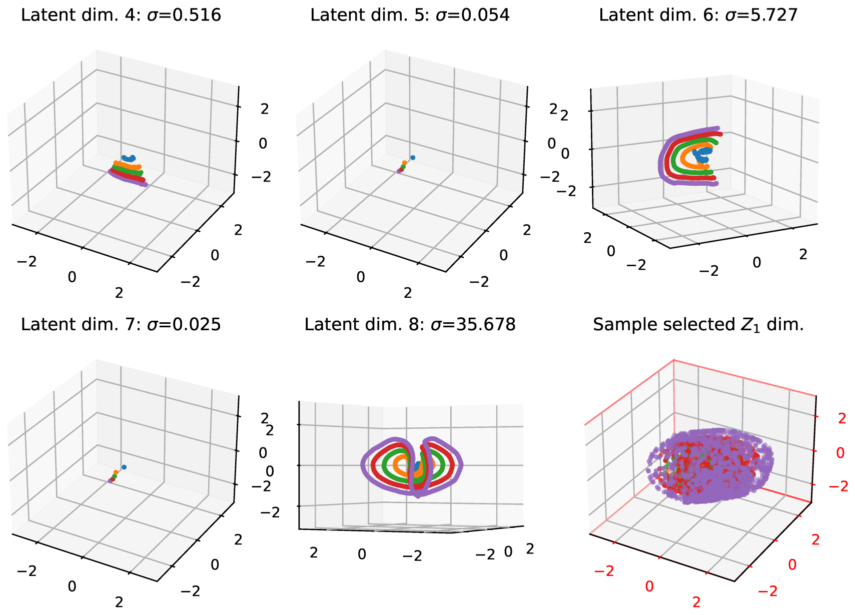


Figure A.2: Illustration of latent traversal in our model for the ellipsoid experiment and latent dimensions $4-8$ in the original input space ($d_x = 3$) for fixed values in the property space dimension 1 (different colours). The selected dimension 6 represents the polar angle ϑ, while dimension 8 can be related to the azimuth angle φ. The last plot (red borders) samples in all selected dimensions (i.e. 6 and 8) which reconstructs the full ellipsoid. We intentionally did not sample the ellipsoid surfaces completely to allow seeing surfaces underneath.

B FEATURE LEARNING AND RANDOM FEATURES IN FINITE DEEP NEURAL NETWORKS

B.1 PARAMETERS AT DIFFERENT WIDTHS

In our investigations, we build on standard PyTorch implementations of LeNet (LeCun et al., 1998) and AlexNet (Krizhevsky, 2014). In the neural tangents framework (see Section 6.5), we reimplement the same LeNet architecture and additionally consider Last-Layer-LeNet in which all parameters are fixed at initialisation except for the weights of the last layer. With increasing width, the number of trainable parameters increases, which is summarised in Appendix Table B.1 for LeNet and Last-Layer-LeNet and in Appendix Table B.2 for AlexNet.

Table B.1: Trainable parameters in LeNet×*Factor* and Last-Layer-LeNet×*Factor* for greyscale input images. RGB input images, i.e. three channel instead of one channel inputs, have a marginal effect on the number of parameters.

Factor	LeNet	Last-Layer-LeNet
1	60 074	850
2	238 026	1690
5	1 479 210	4210
10	5 905 610	8410
25	36 868 010	21 010
60	212 265 610	50 410

B.2 EARLY TRAINING TRAJECTORY

Following Lee et al. (2019), we study training trajectories for data samples $\boldsymbol{x}$ of the MNIST test set during training. For illustration, we plot the iteration $t \in \{0, 1, 2, \ldots\}$ against the standard LeNet output $f^i(\boldsymbol{x}, \boldsymbol{w}_t)$ and the linearisation $f^i_{\text{lin}}(\boldsymbol{x}, \boldsymbol{u}_t)$ for different widths. Note that $\boldsymbol{w}_t$ and $\boldsymbol{u}_t$ are the weights

Table B.2: Trainable parameters in AlexNet×*Factor* for RGB input images.

Factor	AlexNet
1	57 044 810
2	228 032 138
3	512 961 994
4	911 834 378

after t gradient updates for LeNet and LinLeNet trained on MNIST. As we use one-hot encoding, output $i \in \{1, \ldots, C\}$ denotes the predicted output for the correct class of the data point x. The same hyperparameters as for the other MNIST experiments are used (see Section 6.4.1). A fixed random seed ensures that both LeNet and LinLeNet at a particular width factor are initialised exactly the same and receive the same mini-batches during training. Exemplary results are shown in Figure B.1 for a digit 8 of the test set, with similar results being obtained for other samples, too. At small widths, training trajectories immediately diverge. With increasing width, the curves behave more similar; they are however not close in a path-wise sense, but the statistics of training trajectories become more alike.

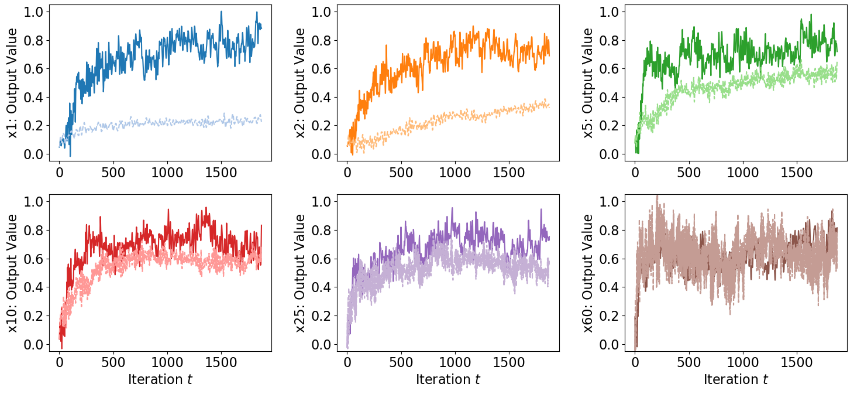


Figure B.1: Training trajectories of LeNet (dark) and LinLeNet (light) do not stay close for small width factors. Shown are the output values during training for the same MNIST input example from the validation set at different widths.

B.3 EFFECTIVE RANK

The effective rank was introduced by Roy and Vetterli (2007) and can be viewed as the exponential entropy of normalised singular values. We restate the main definition in the following.

Definition 7 (Effective Rank) *Let A be a complex-valued non-all-zero matrix of size $M \times N$ with (real positive) singular values $\sigma_1 \geqslant \sigma_2 \geqslant \ldots \geqslant \sigma_Q \geqslant 0$, where $Q = \min\{M, N\}$. Let $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_Q)^\top$ and the singular value distribution be*

$$p_k = \frac{\sigma_k}{\sum_{j=1}^{Q} \sigma_j} \quad \text{with } k = 1, 2, \ldots, Q. \tag{B.1}$$

The effective rank of matrix A is then defined as

$$erank(A) := \exp\left(H(p_1, p_2, \ldots, p_Q)\right) \tag{B.2}$$

where $H(p_1, p_2, \ldots, p_Q)$ is the Shannon entropy

$$H(p_1, p_2, \ldots, p_Q) = -\sum_{k=1}^{Q} p_k \log p_k. \tag{B.3}$$

In comparison to the usual notion of rank, an important property of the effective rank is that $erank(A) \leqslant rank(A)$ (Roy and Vetterli, 2007).

B.4 ACCURACY RESULTS IN FINITE-WIDTH ANALYSIS

Tables B.3 and B.4 provide the values for train and test accuracy in the LeNet and AlexNet experiments, respectively. The sample mean accuracy as well as sample standard deviation for five independent reruns of models are shown.

Table B.3: Mean accuracy and standard deviation for five independent reruns of the LeNet experiments.

		×1	×2	×5	×10	×25	×60
MNIST (see Figure 6.5)							
Test	Lin.	94.48 ± 1.04	96.42 ± 0.17	97.86 ± 0.02	98.36 ± 0.04	98.72 ± 0.05	98.91 ± 0.1
	LeNet	99.15 ± 0.1	99.29 ± 0.05	99.38 ± 0.06	99.4 ± 0.03	99.42 ± 0.04	99.39 ± 0.06
Train	Lin.	94.3 ± 1.07	96.56 ± 0.33	98.14 ± 0.07	98.82 ± 0.05	99.45 ± 0.02	99.83 ± 0.03
	LeNet	99.86 ± 0.02	99.94 ± 0.01	99.95 ± 0.01	99.97 ± 0.00	99.97 ± 0.01	99.97 ± 0.01
MNIST with translation (see Figure 6.6)							
Test	Lin.	69.95 ± 5.86	80.91 ± 1.77	89.57 ± 0.49	92.13 ± 0.11	94.25 ± 0.48	94.66 ± 0.58
	LeNet	97.6 ± 0.16	98.35 ± 0.12	98.61 ± 0.03	98.63 ± 0.13	98.63 ± 0.33	98.57 ± 0.12
Train	Lin.	68.93 ± 5.44	80.54 ± 1.64	88.89 ± 0.29	91.76 ± 0.13	94.0 ± 0.1	94.42 ± 0.29
	LeNet	97.55 ± 0.05	98.28 ± 0.06	98.59 ± 0.03	98.71 ± 0.05	98.76 ± 0.05	98.77 ± 0.03
CIFAR-10 (see Figure 6.7)							
Test	Lin.	42.98 ± 0.53	48.07 ± 1.12	54.42 ± 0.4	58.23 ± 0.43	62.47 ± 0.26	65.8 ± 0.23
	LeNet	63.2 ± 0.58	69.58 ± 0.48	75.76 ± 0.22	77.56 ± 0.21	78.83 ± 0.1	78.97 ± 0.13
Train	Lin.	43.99 ± 0.97	50.3 ± 1.52	60.45 ± 1.14	68.32 ± 0.92	81.24 ± 0.74	93.84 ± 0.33
	LeNet	92.07 ± 0.24	98.53 ± 0.07	99.76 ± 0.04	99.92 ± 0.01	99.96 ± 0.00	99.98 ± 0.00

Table B.4: Mean accuracy and standard deviation for five independent reruns of the AlexNet experiment (see Figure 6.8).

		×1	×2	×3	×4
Test	Lin. 0.1	30.48 ± 1.68	33.52 ± 0.63	33.48 ± 1.14	33.88 ± 1.68
	Lin. 1.0	33.64 ± 1.26	35.2 ± 1.53	36.76 ± 1.52	36.6 ± 2.05
	AlexNet	54.04 ± 1.13	54.72 ± 1.27	54.72 ± 0.98	55.16 ± 1.54
Train	Lin. 0.1	36.75 ± 0.5	42.62 ± 0.8	46.64 ± 0.54	49.84 ± 0.95
	Lin. 1.0	51.05 ± 2.17	64.45 ± 3.12	72.63 ± 3.02	79.45 ± 3.12
	AlexNet	99.15 ± 0.05	99.19 ± 0.03	99.16 ± 0.05	99.16 ± 0.04

B.5 ACCURACY RESULTS IN INFINITE-WIDTH ANALYSIS

Tables B.5 (MNIST), B.6 (shifted MNIST), B.7 (shifted-resized MNIST), and B.8 (CIFAR-10) provide the values for train and test accuracy in the LeNet$_{\text{nt}}$ experiments. The sample mean accuracy as well as sample standard deviation for five independent reruns of models are shown. Additionally, Figures B.2 and B.3 provide rescaled visualisations of Figure 6.12 (all MNIST variants) and Figure B.4 provides a rescaled visualisation of Figure 6.13 (CIFAR-10).

Table B.5: Accuracy of LeNet$_{\text{nt}}$, LinLeNet$_{\text{nt}}$ (Lin.), and Last-Layer-LeNet (LL) on 50k-MNIST at different widths. The infinite-width kernels achieve a test accuracy of 99.40% (NNGP) and 99.20% (NTK). Networks exceeding the NTK test performance are indicated by $*$. See Figure 6.12 and Appendix Figures B.2 as well as B.3 for visualisations of the reported values.

		×1	**×2**	**×5**	**×10**	**×25**	**×60**
Test	LL	85.53 ± 0.52	90.75 ± 0.48	95.49 ± 0.26	97.07 ± 0.10	98.15 ± 0.10	98.71 ± 0.06
	Lin.	97.90 ± 0.14	98.55 ± 0.07	98.91 ± 0.06	99.06 ± 0.03	99.16 ± 0.02	99.19 ± 0.07
	LeNet	98.70 ± 0.09	99.04 ± 0.07	$99.30^* \pm 0.07$	$99.36^* \pm 0.07$	$99.35^* \pm 0.02$	$99.26^* \pm 0.05$
Train	LL	84.77 ± 0.63	90.34 ± 0.48	95.38 ± 0.23	97.05 ± 0.11	98.46 ± 0.03	99.17 ± 0.04
	Lin.	98.44 ± 0.09	99.33 ± 0.06	99.81 ± 0.02	99.89 ± 0.01	99.96 ± 0.01	99.98 ± 0.00
	LeNet	99.94 ± 0.01	99.98 ± 0.01	99.99 ± 0.00	100.0 ± 0.00	100.0 ± 0.00	100.0 ± 0.00

Table B.6: Accuracy of LeNet$_{\text{nt}}$, LinLeNet$_{\text{nt}}$ (Lin.), and Last-Layer-LeNet (LL) on shifted 50k-MNIST at different widths. The infinite-width kernels achieve a test accuracy of 98.40% (NNGP) and 98.21% (NTK). Networks exceeding the test performance of the NTK are indicated by $*$ and of the NNGP kernel by $**$. See Fig 6.12 and Appendix Figures B.2 as well as B.3 for visualisations of the reported values.

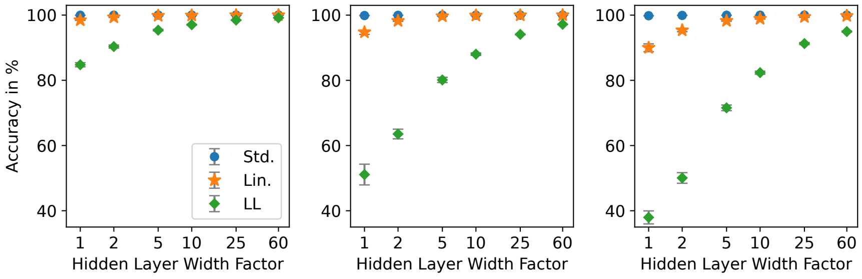
		×1	**×2**	**×5**	**×10**	**×25**	**×60**
Test	LL	51.73 ± 2.62	63.83 ± 1.56	79.59 ± 0.69	87.41 ± 0.46	92.67 ± 0.12	95.61 ± 0.18
	Lin.	92.43 ± 0.45	95.38 ± 0.27	97.10 ± 0.11	97.62 ± 0.04	97.76 ± 0.07	97.95 ± 0.06
	LeNet	97.21 ± 0.17	$98.25^* \pm 0.07$	98.15 ± 0.44	$98.36^* \pm 0.14$	$98.5^{**} \pm 0.04$	98.07 ± 0.10
Train	LL	51.10 ± 3.18	63.54 ± 1.48	80.12 ± 0.80	88.02 ± 0.38	94.09 ± 0.10	97.19 ± 0.06
	Lin.	94.75 ± 0.66	98.24 ± 0.13	99.69 ± 0.06	99.89 ± 0.02	99.96 ± 0.01	99.99 ± 0.00
	LeNet	99.88 ± 0.01	99.94 ± 0.01	99.98 ± 0.01	99.99 ± 0.01	100.0 ± 0.00	100.0 ± 0.00

Table B.7: Accuracy of LeNet$_{nt}$, LinLeNet$_{nt}$ (Lin.), and Last-Layer-LeNet (LL) on shifted-resized 50k-MNIST at different widths. The infinite-width kernels achieve a test accuracy of 98.21% (NNGP) and 97.47% (NTK). Networks exceeding the NTK test performance are indicated by $*$. See Figure 6.12 and Appendix Figures B.2 as well as B.3 for visualisations of the reported values.

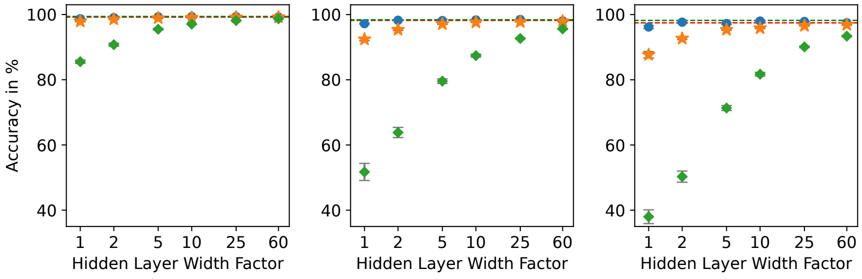
		×**1**	×**2**	×**5**	×**10**	×**25**	×**60**
Test	LL	37.99 ± 2.11	50.28 ± 1.72	71.33 ± 0.77	81.70 ± 0.57	90.09 ± 0.23	93.39 ± 0.20
	Lin.	87.68 ± 1.00	92.66 ± 0.32	95.36 ± 0.20	95.85 ± 0.15	96.58 ± 0.12	96.89 ± 0.18
	LeNet	96.18 ± 0.44	$97.67^* \pm 0.38$	97.26 ± 0.30	$98.02^* \pm 0.27$	$97.88^* \pm 0.09$	$97.49^* \pm 0.03$
Train	LL	37.96 ± 2.00	50.06 ± 1.63	71.56 ± 0.87	82.35 ± 0.48	91.28 ± 0.32	94.96 ± 0.18
	Lin.	89.99 ± 1.20	95.36 ± 0.43	98.25 ± 0.12	98.89 ± 0.10	99.46 ± 0.12	99.77 ± 0.04
	LeNet	99.82 ± 0.02	99.92 ± 0.02	99.96 ± 0.01	99.98 ± 0.01	100.0 ± 0.00	100.0 ± 0.00

Table B.8: Accuracy of LeNet$_{nt}$, LinLeNet$_{nt}$ (Lin.), and Last-Layer-LeNet (LL) on CIFAR-10 at different widths. The infinite-width kernels achieve a test accuracy of 69.53% (NNGP) and 68.77% (NTK). None of the considered networks exceed the kernel test performance. See Figure 6.13 and Appendix Figure B.4 for visualisations of the reported values.

		×**1**	×**2**	×**5**	×**10**	×**25**	×**60**
Test	LL	30.62 ± 0.80	35.55 ± 0.56	42.04 ± 0.25	47.10 ± 0.38	51.91 ± 0.33	56.24 ± 0.53
	Lin.	47.88 ± 0.74	53.45 ± 0.70	58.52 ± 1.05	60.65 ± 0.45	61.93 ± 0.74	62.42 ± 0.60
	LeNet	51.72 ± 1.90	58.48 ± 1.12	60.37 ± 1.22	60.73 ± 0.77	60.04 ± 0.65	58.31 ± 0.80
Train	LL	30.80 ± 0.78	36.09 ± 0.69	43.75 ± 0.29	49.49 ± 0.18	58.19 ± 0.41	68.41 ± 0.20
	Lin.	63.91 ± 1.11	82.64 ± 0.70	96.66 ± 1.03	99.40 ± 0.15	99.90 ± 0.05	99.98 ± 0.00
	LeNet	95.17 ± 0.48	99.30 ± 0.08	99.90 ± 0.02	100.0 ± 0.00	100.0 ± 0.00	100.0 ± 0.00

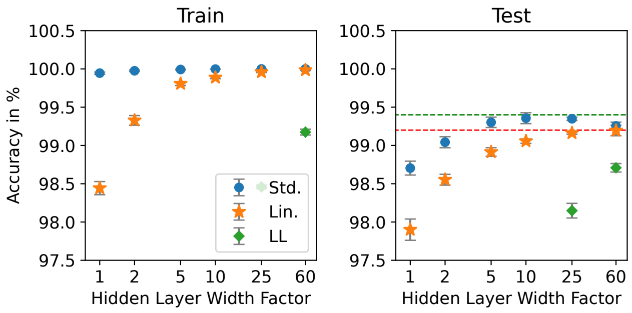


(a) Train accuracy.

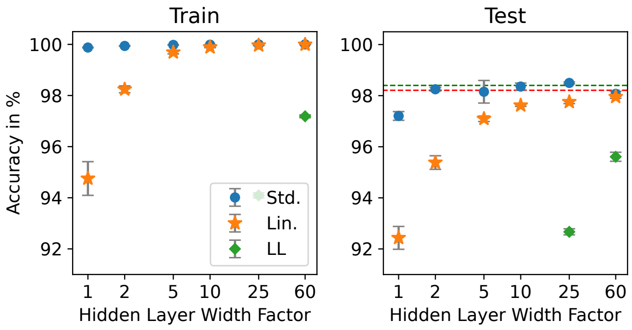


(b) Test accuracy.

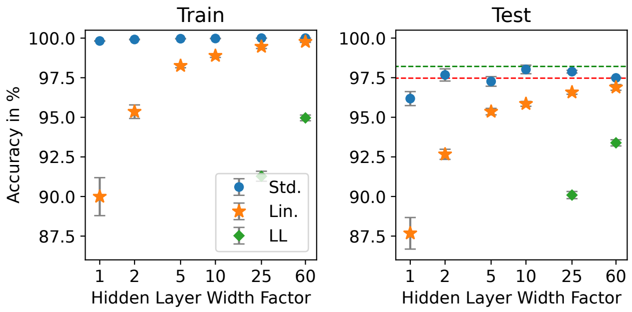
Figure B.2: Accuracy of LeNet$_{\text{nt}}$ (•), LinLeNet$_{\text{nt}}$ (⋆), and Last-Layer-LeNet (⬥) trained on MNIST at different widths (values in Appendix Tables B.5, B.6, B.7). Dashed lines indicate the result for the infinite-width case for the NTK (red) and NNGP kernel (green). The first column shows results for the standard MNIST dataset, the second column for shifted MNIST, and the third column for shifted-resized MNIST. This visualisation includes all results for Last-Layer-LeNet at all widths. Visualisations with different y-axis scaling can be found in Figure 6.12 and Appendix Figure B.3.



(a) Standard MNIST.



(b) Shifted MNIST.



(c) Shifted-resized MNIST.

Figure B.3: Accuracy of LeNet$_{\text{nt}}$ (•), LinLeNet$_{\text{nt}}$ (⋆), and Last-Layer-LeNet (♦) trained on MNIST at different widths (values in Appendix Tables B.5, B.6, B.7). Dashed lines indicate the result for the infinite-width case for the NTK (red) and NNGP kernel (green). This visualisation focuses on the results for LeNet and LinLeNet on the different datasets. Visualisations with different y-axis scaling can be found in Figure 6.12 and Appendix Figure B.2.

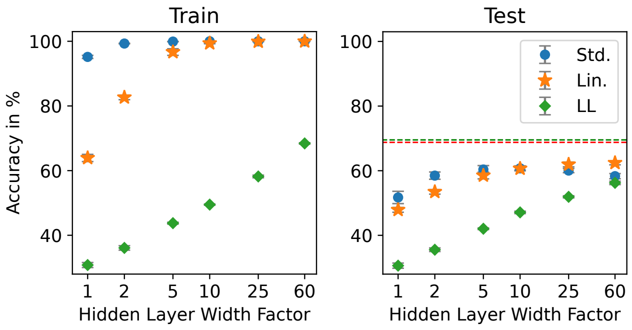


Figure B.4: Accuracy of LeNet$_{nt}$ (●), LinLeNet$_{nt}$ (⋆), and Last-Layer-LeNet (◆) trained on CIFAR-10 at different widths (values in Appendix Table B.8). Dashed lines indicate the result for the infinite-width case for the NTK (red) and NNGP kernel (green). This visualisation includes all results for Last-Layer-LeNet at all widths. The visualisation in the main text focuses on the results for LeNet and LinLeNet (see Figure 6.13).

B.6 RESULTS FOR MNIST WITH RANDOM LABELS

In Section 6.5.3 we perform an additional MNIST experiment in which the relation between input image and label is removed. To this end, we randomly generated a new label between 0 and 9 for each input image. Figure 6.14 visualises these results. The numerical values for the sample mean accuracy as well as sample standard deviation for five independent reruns of LeNet$_{nt}$, LinLeNet$_{nt}$, and Last-Layer-LeNet are provided in Appendix Tables B.9 (MNIST with random labels), B.10 (shifted MNIST with random labels), B.11 (shifted-resized MNIST with random labels). Note that in all settings the same random labels were used.

Table B.9: Accuracy of LeNet$_{\text{nt}}$, LinLeNet$_{\text{nt}}$ (Lin.), and Last-Layer-LeNet (LL) on MNIST with random labels at different widths. The infinite-width kernels achieve a test accuracy of 9.75% (NNGP) and 10.05% (NTK). None of the considered networks exceed the kernel test performance. See Appendix Figure 6.14 for visualisations of the reported values.

		×1	×2	×5	×10	×25	×60
Test	LL	10.01 ± 0.17	9.92 ± 0.23	10.13 ± 0.12	10.13 ± 0.25	10.08 ± 0.52	9.89 ± 0.48
	Lin.	10.03 ± 0.50	10.21 ± 0.37	10.05 ± 0.29	10.15 ± 0.18	9.90 ± 0.11	9.67 ± 0.27
	LeNet	9.98 ± 0.25	9.99 ± 0.47	10.06 ± 0.25	10.11 ± 0.34	10.09 ± 0.50	9.75 ± 0.38
Train	LL	11.94 ± 0.11	12.81 ± 0.20	14.74 ± 0.14	17.04 ± 0.16	22.21 ± 0.04	30.55 ± 0.21
	Lin.	28.71 ± 1.80	44.15 ± 1.71	67.49 ± 1.95	78.57 ± 1.75	88.74 ± 1.85	95.28 ± 0.48
	LeNet	57.52 ± 1.11	95.99 ± 1.11	100.00 ± 0.00	100.0 ± 0.00	100.0 ± 0.00	100.0 ± 0.00

Table B.10: Accuracy of LeNet$_{\text{nt}}$, LinLeNet$_{\text{nt}}$ (Lin.), and Last-Layer-LeNet (LL) on shifted MNIST with random labels at different widths. The infinite-width kernels achieve a test accuracy of 9.68% (NNGP) and 9.45% (NTK). None of the considered networks exceed the kernel test performance. See Appendix Figure 6.14 for visualisations of the reported values.

		×1	×2	×5	×10	×25	×60
Test	LL	10.05 ± 0.42	10.09 ± 0.24	9.99 ± 0.12	9.87 ± 0.33	10.09 ± 0.14	9.73 ± 0.35
	Lin.	10.22 ± 0.36	10.01 ± 0.08	9.96 ± 0.21	10.00 ± 0.41	9.85 ± 0.48	10.08 ± 0.21
	LeNet	10.16 ± 0.3	9.95 ± 0.42	10.07 ± 0.36	9.92 ± 0.39	10.02 ± 0.21	9.99 ± 0.27
Train	LL	12.06 ± 0.04	12.82 ± 0.10	14.97 ± 0.16	17.28 ± 0.30	22.91 ± 0.14	32.55 ± 0.24
	Lin.	31.96 ± 1.75	51.56 ± 1.74	79.87 ± 2.65	90.63 ± 1.74	96.71 ± 0.91	99.21 ± 0.11
	LeNet	60.02 ± 1.29	97.38 ± 0.66	100.00 ± 0.00	100.0 ± 0.00	100.0 ± 0.00	100.0 ± 0.00

Table B.11: Accuracy of LeNet$_{\text{nt}}$, LinLeNet$_{\text{nt}}$ (Lin.), and Last-Layer-LeNet (LL) on shifted-resized MNIST with random labels at different widths. The infinite-width kernels achieve a test accuracy of 9.95% (NNGP) and 10.01% (NTK). None of the considered networks exceed the kernel test performance. See Appendix Figure 6.14 for visualisations of the reported values.

		×1	×2	×5	×10	×25	×60
Test	LL	9.89 ± 0.40	10.15 ± 0.54	10.06 ± 0.42	10.05 ± 0.12	9.99 ± 0.37	9.75 ± 0.21
	Lin.	10.02 ± 0.43	9.96 ± 0.33	9.91 ± 0.41	9.80 ± 0.28	10.07 ± 0.15	9.94 ± 0.52
	LeNet	9.86 ± 0.41	9.72 ± 0.34	10.08 ± 0.22	9.97 ± 0.30	10.14 ± 0.17	10.07 ± 0.31
Train	LL	11.95 ± 0.17	12.80 ± 0.03	14.80 ± 0.21	17.08 ± 0.18	21.85 ± 0.33	28.33 ± 0.30
	Lin.	27.1 ± 1.64	38.48 ± 2.12	56.83 ± 2.42	66.75 ± 2.82	78.90 ± 2.92	88.77 ± 1.01
	LeNet	56.56 ± 1.26	94.58 ± 0.75	100.00 ± 0.00	100.0 ± 0.00	100.0 ± 0.00	100.0 ± 0.00

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

BIBLIOGRAPHY

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, software available from tensorflow.org, 2015. ↑59, ↑97

Achille, A. and Soatto, S.: Information dropout: Learning optimal representations through noisy computation, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018. ↑90

Advani, M. S., Saxe, A. M., and Sompolinsky, H.: High-dimensional dynamics of generalization error in neural networks, Neural Networks, 132, 428–446, 2020. ↑26

Ainsworth, S. K., Foti, N. J., Lee, A. K. C., and Fox, E. B.: oi-VAE: Output Interpretable VAEs for Nonlinear Group Factor Analysis, in: Proceedings of the 35th International Conference on Machine Learning, 2018. ↑91

Akeret, J., Chang, C., Lucchi, A., and Refregier, A.: Radio frequency interference mitigation using deep convolutional neural networks, Astron. Comput., 18, 35–39, 2017. ↑59

Alder, S., Prasuhn, V., Liniger, H., Herweg, K., Hurni, H., Candinas, A., and Gujer, H. U.: A high-resolution map of direct and indirect connectivity of erosion risk areas to surface waters in Switzerland-A risk assessment tool for planning and policy-making, Land Use Policy, 48, 236–249, 2015. ↑49

Alekseev, A. and Bobe, A.: GaborNet: Gabor filters with learnable parameters in deep convolutional neural network, in: 2019 International Conference on Engineering and Telecommunication (EnT), pp. 1–4, IEEE, 2019. ↑119

Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K.: Deep Variational Information Bottleneck, in: 5th International Conference on Learning Representations, 2017. ↑41, ↑42, ↑88, ↑89, ↑90, ↑91

Alemohammad, S., Wang, Z., Balestriero, R., and Baraniuk, R.: The recurrent neural tangent kernel, arXiv preprint arXiv:2006.10246, 2020. ↑117

Alewell, C., Meusburger, K., Brodbeck, M., and Bänninger, D.: Methods to describe and predict soil erosion in mountain regions, Landsc. Urban Plan., 88, 46–53, 2008. ↑62

Alewell, C., Egli, M., and Meusburger, K.: An attempt to estimate tolerable soil erosion rates by matching soil formation with denudation in Alpine grasslands, J. Soils Sediments, 15, 1383–1399, 2015. ↑52

Alewell, C., Borrelli, P., Meusburger, K., and Panagos, P.: Using the USLE: Chances, challenges and limitations of soil erosion modelling, Int. Soil Water Conserv. Res., 7, 203–225, 2019. ↑49

Allen-Zhu, Z., Li, Y., and Song, Z.: A convergence theory for deep learning via over-parameterization, in: International Conference on Machine Learning, pp. 242–252, PMLR, 2019. ↑119

Alshaikhli, T., Liu, W., and Maruyama, Y.: Automated method of road extraction from aerial images using a deep convolutional neural network, Appl. Sci., 9, 4825, 2019. ↑50

Anderson, E., Bai, Z., Bischof, C., Blackford, L. S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D.: LAPACK Users' guide, SIAM, 1999. ↑132

Apollo, M., Andreychouk, V., and Bhattarai, S. S.: Short-term impacts of livestock grazing on vegetation and track formation in a high mountain environment: A case study from the Himalayan Miyar Valley (India), Sustainability, 10, 951, 2018. ↑48

Armato, S. G., McLennan, G., Bidaut, L., McNitt-Gray, M. F., Meyer, C. R., Reeves, A. P., Zhao, B., Aberle, D. R., Henschke, C. I., Hoffman, E. A., et al.: The lung image database consortium (LIDC) and image database resource initiative (IDRI): a completed reference database of lung nodules on CT scans, Medical physics, 38, 915–931, 2011. ↑146

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R. R., and Wang, R.: On exact computation with an infinitely wide neural net, Advances in Neural Information Processing Systems, 32, 2019. ↑34, ↑117, ↑120, ↑141

Bai, S., Kolter, J. Z., and Koltun, V.: Deep equilibrium models, Advances in Neural Information Processing Systems, 32, 2019. ↑119

Baumhoer, C. A., Dietz, A. J., Kneisel, C., and Kuenzer, C.: Automated extraction of antarctic glacier and ice shelf fronts from Sentinel-1 imagery using deep learning, Remote Sens., 11, 2529, 2019. ↑50, ↑64

Belkin, M., Hsu, D., Ma, S., and Mandal, S.: Reconciling modern machine-learning practice and the classical bias–variance trade-off, Proceedings of the National Academy of Sciences, 116, 15 849–15 854, 2019. ↑26

Belkin, M., Hsu, D., and Xu, J.: Two Models of Double Descent for Weak Features, SIAM Journal on Mathematics of Data Science, 2, 1167–1180, 2020. ↑27

Bengio, Y., Courville, A., and Vincent, P.: Representation learning: A review and new perspectives, IEEE transactions on pattern analysis and machine intelligence, 35, 1798–1828, 2013. ↑150

Bhatia, M.: Modeling Ambiguity in Semantic Segmentation, Master's thesis, University of Basel, 2020. ↑146

Bircher, P., Liniger, H. P., and Prasuhn, V.: Comparing different multiple flow algorithms to calculate RUSLE factors of slope length (L) and slope steepness (S) in Switzerland, Geomorphology, 346, 106 850, 2019. ↑49

Bishop, C. M.: Pattern recognition and machine learning, vol. 4, Springer, 2006. ↑13, ↑21

Bouchacourt, D., Tomioka, R., and Nowozin, S.: Multi-Level Variational Autoencoder: Learning Disentangled Representations From Grouped Observations, in: AAAI Conference on Artificial Intelligence, 2018. ↑88, ↑90

Breiman, L.: Statistical modeling: The two cultures, Statistical Science, 16, 199–231, 2001. ↑2

Broomhead, D. S. and Lowe, D.: Radial basis functions, multi-variable functional interpolation and adaptive networks, Tech. rep., Royal Signals and Radar Establishment Malvern (United Kingdom), 1988. ↑148

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D.: Language models are few-shot learners, Advances in neural information processing systems, 33, 1877–1901, 2020. ↑3, ↑26

Bryson, A. and Ho, Y.: Applied optimal control: optimization, estimation, and control, Blaisdell Pub. Co., 1969. ↑24

Bundzel, M., Jaščur, M., Kováč, M., Lieskovský, T., Sinčák, P., and Tkáčik, T.: Semantic segmentation of airborne lidar data in maya archaeology, Remote Sensing, 12, 1–22, 2020. ↑50

CH2018: CH2018 – Climate Scenarios for Switzerland, Tech. rep., National Centre for Climate Services, Zurich, 2018. ↑9, ↑10

Chechik, G., Globerson, A., Tishby, N., and Weiss, Y.: Information bottleneck for Gaussian variables, in: Journal of Machine Learning Research, 2005. ↑41, ↑93

Chen, H., Engkvist, O., Wang, Y., Olivecrona, M., and Blaschke, T.: The rise of deep learning in drug discovery, Drug discovery today, 23, 1241–1250, 2018a. ↑3

Chen, J., Lu, Y., Yu, Q., Luo, X., Adeli, E., Wang, Y., Lu, L., Yuille, A. L., and Zhou, Y.: Transunet: Transformers make strong encoders for medical image segmentation, arXiv preprint arXiv:2102.04306, 2021. ↑46

Chen, R. T., Li, X., Grosse, R., and Duvenaud, D.: Isolating sources of disentanglement in variational autoencoders, arXiv preprint arXiv:1802.04942, 2018b. ↑88, ↑90

Chen, S., He, H., and Su, W. J.: Label-Aware Neural Tangent Kernel: Toward Better Generalization and Local Elasticity, in: NeurIPS, 2020. ↑116

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P.: Infogan: Interpretable representation learning by information maximizing generative adversarial nets, arXiv preprint arXiv:1606.03657, 2016. ↑88, ↑90

Chicharro, D., Besserve, M., and Panzeri, S.: Causal learning with sufficient statistics: an information bottleneck approach, arXiv preprint arXiv:2010.05375, 2020. ↑90

Chizat, L. and Bach, F.: On the global convergence of gradient descent for over-parameterized models using optimal transport, in: Advances in neural information processing systems, pp. 3036–3046, 2018. ↑142

Chizat, L., Oyallon, E., and Bach, F.: On lazy training in differentiable programming, in: Advances in Neural Information Processing Systems, pp. 2933–2943, 2019. ↑116, ↑117, ↑120, ↑121, ↑141

Çiçek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., and Ronneberger, O.: 3D U-Net: learning dense volumetric segmentation from sparse annotation, in: International conference on medical image computing and computer-assisted intervention, pp. 424–432, Springer, 2016. ↑45

Creswell, A., Mohamied, Y., Sengupta, B., and Bharath, A. A.: Adversarial Information Factorization, 2018. ↑90

Cybenko, G.: Approximation by superpositions of a sigmoidal function, Mathematics of control, signals and systems, 2, 303–314, 1989. ↑3

de G. Matthews, A. G., Rowland, M., Hron, J., Turner, R. E., and Ghahramani, Z.: Gaussian Process Behaviour in Wide Deep Neural Networks, 2018. ↑28, ↑116, ↑121

Decaëns, T., Jiménez, J., Gioia, C., Measey, G., and Lavelle, P.: The values of soil animals for conservation biology, European Journal of Soil Biology, 42, S23–S38, iCSZ, 2006. ↑9

Dechesne, C., Lassalle, P., and Lefèvre, S.: Bayesian U-Net: Estimating uncertainty in semantic segmentation of earth observation images, Remote Sensing, 13, 3836, 2021. ↑46, ↑145

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805, 2018. ↑3, ↑26

D'Oleire-Oltmanns, S., Marzolff, I., Peter, K. D., and Ries, J. B.: Unmanned aerial vehicle (UAV) for monitoring soil erosion in Morocco, Remote Sens., 4, 3390–3416, 2012. ↑49

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N.: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, in: International Conference on Learning Representations, 2021. ↑3, ↑26, ↑43

Du, S. S., Zhai, X., Poczos, B., and Singh, A.: Gradient Descent Provably Optimizes Over-parameterized Neural Networks, in: International Conference on Learning Representations, 2019. ↑117, ↑119

Durán, J. M. and Jongsma, K. R.: Who is afraid of black box algorithms? On the epistemological and ethical basis of trust in medical AI, Journal of Medical Ethics, 47, 329–335, 2021. ↑3

EEA: Regional climate change and adaptation — The Alps facing the challenge of changing water resources, Tech. Rep. 8, European Environmental Agency, 2009. ↑48

Eisank, C., Hölbling, D., Friedl, B., and Chin, Y.: Expert knowledge for object-based landslide mapping in Taiwan, South-Eastern Eur. J. Earth Observ., 3, 347–350, 2014. ↑49

Fischer, F. K., Kistler, M., Brandhuber, R., Maier, H., Treisch, M., and Auerswald, K.: Validation of official erosion modelling based on high-resolution radar rain data by aerial photo erosion classification, Earth Surf. Proc. Land, 43, 187–194, 2018. ↑49

Flood, N., Watson, F., and Collett, L.: Using a U-net convolutional neural network to map woody vegetation extent from high resolution satellite imagery across Queensland, Australia, Int. J. Appl. Earth Obs., 82, 101 897, 2019. ↑50

FOA: Status of the world's soil resources: main report., Tech. rep., UN Food and Agriculture Organization, Rome, 2015. ↑9

Fort, S., Dziugaite, G. K., Paul, M., Kharaghani, S., Roy, D. M., and Ganguli, S.: Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel, Advances in Neural Information Processing Systems, 33, 5850–5861, 2020. ↑142

Fu, Y., Liu, K., Shen, Z., Deng, J., Gan, M., Liu, X., Lu, D., and Wang, K.: Mapping impervious surfaces in town-rural transition belts using China's

GF-2 imagery and object-based deep CNNs, Remote Sens., 11, 280, 2019. ↑64, ↑71

Fuhrer, J., Beniston, M., Fischlin, A., Frei, C., Goyette, S., Jasper, K., and Pfister, C.: Climate risks and their impact on agriculture and forests in Switzerland, Clim. Change, 79, 79–102, 2006. ↑9, ↑48

Fukushima, K.: Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position, Biological Cybernetics, 36, 193–202, 1980. ↑43

Garriga-Alonso, A., Rasmussen, C. E., and Aitchison, L.: Deep Convolutional Networks as shallow Gaussian Processes, in: International Conference on Learning Representations, 2019. ↑28, ↑116

Geiger, M., Spigler, S., Jacot, A., and Wyart, M.: Disentangling feature and lazy training in deep neural networks, Journal of Statistical Mechanics: Theory and Experiment, 2020, 113 301, 2020. ↑121

Geman, S., Bienenstock, E., and Doursat, R.: Neural networks and the bias/variance dilemma, Neural computation, 4, 1–58, 1992. ↑15

Ghorbanzadeh, O., Blaschke, T., Gholamnia, K., Meena, S. R., Tiede, D., and Aryal, J.: Evaluation of different machine learning methods and deep-learning convolutional neural networks for landslide detection, Remote Sens., 11, 2019. ↑71

Girshick, R., Donahue, J., Darrell, T., and Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580–587, 2014. ↑43

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A.: Automatic chemical design using a data-driven continuous representation of molecules, ACS central science, 4, 268–276, 2018. ↑3, ↑42, ↑89, ↑92, ↑99, ↑105

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y.: Generative adversarial networks, Communications of the ACM, 63, 139–144, 2020. ↑36

Graves, A., Mohamed, A.-r., and Hinton, G.: Speech recognition with deep recurrent neural networks, in: 2013 IEEE international conference on acoustics, speech and signal processing, pp. 6645–6649, Ieee, 2013. ↑3

Guirado, E., Tabik, S., Alcaraz-Segura, D., Cabello, J., and Herrera, F.: Deep-learning Versus OBIA for scattered shrub detection with Google Earth Imagery: Ziziphus lotus as case study, Remote Sens., 9, 1220, 2017. ↑64, ↑71

Guzzetti, F., Mondini, A. C., Cardinali, M., Fiorucci, F., Santangelo, M., and Chang, K. T.: Landslide inventory maps: New tools for an old problem, Earth-Sci. Rev., 112, 42–66, 2012. ↑49

Hamdi, Z. M., Brandmeier, M., and Straub, C.: Forest damage assessment using deep learning on high resolution remote sensing data, Remote Sens., 11, 1976, 2019. ↑50, ↑69

Han, Y. and Ye, J. C.: Framing U-Net via deep convolutional framelets: Application to sparse-view CT, IEEE transactions on medical imaging, 37, 1418–1429, 2018. ↑45

Hanin, B. and Nica, M.: Finite Depth and Width Corrections to the Neural Tangent Kernel, in: International Conference on Learning Representations, 2020. ↑116, ↑119

Hansen, K., Biegler, F., Ramakrishnan, R., Pronobis, W., Von Lilienfeld, O. A., Müller, K.-R., and Tkatchenko, A.: Machine learning predictions of molecular properties: Accurate many-body potentials and nonlocality in chemical space, The journal of physical chemistry letters, 6, 2326–2331, 2015. ↑105

Hastie, T., Montanari, A., Rosset, S., and Tibshirani, R. J.: Surprises in high-dimensional ridgeless least squares interpolation, The Annals of Statistics, 50, 949–986, 2022. ↑27

Hatamizadeh, A., Tang, Y., Nath, V., Yang, D., Myronenko, A., Landman, B., Roth, H. R., and Xu, D.: Unetr: Transformers for 3d medical image segmentation, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 574–584, 2022. ↑45

He, K., Zhang, X., Ren, S., and Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE international conference on computer vision, pp. 1026–1034, 2015. ↑123

He, K., Zhang, X., Ren, S., and Sun, J.: Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016. ↑43, ↑120

Heydari, S. S. and Mountrakis, G.: Meta-analysis of deep neural networks in remote sensing: A comparative study of mono-temporal classification to support vector machines, ISPRS J. Photogramm. Remote Sens., 152, 192–210, 2019. ↑49

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A.: beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework, in: International Conference on Learning Representations, 2017. ↑39, ↑88, ↑90, ↑91, ↑97, ↑108

Ho, J., Jain, A., and Abbeel, P.: Denoising diffusion probabilistic models, Advances in Neural Information Processing Systems, 33, 6840–6851, 2020. ↑36

Hölbling, D., Friedl, B., and Eisank, C.: An object-based approach for semi-automated landslide change detection and attribution of changes to landslide classes in northern Taiwan, Earth Sci. Inform., 8, 327–335, 2015. ↑49

Hölbling, D., Betts, H., Spiekermann, R., and Phillips, C.: Identifying spatio-temporal landslide hotspots on North Island, New Zealand, by analyzing historical and recent aerial photography, Geosciences, 6, 48, 2016. ↑49

Hölbling, D., Abad, L., Dabiri, Z., Prasicek, G., Tsai, T.-t., and Argentin, A.-l.: Mapping and analyzing the evolution of the Butangbunasi landslide using Landsat time series with respect to heavy rainfall events during Typhoons, Appl. Sci., 10, 630, 2020. ↑49

Hornik, K.: Approximation capabilities of multilayer feedforward networks, Neural networks, 4, 251–257, 1991. ↑3

Hron, J., Bahri, Y., Sohl-Dickstein, J., and Novak, R.: Infinite attention: NNGP and NTK for deep attention networks, in: International Conference on Machine Learning, pp. 4376–4386, PMLR, 2020. ↑28, ↑117

Hu, K., Ren, Z., Siska, D., and Szpruch, L.: Mean-field Langevin dynamics and energy landscape of neural networks, arXiv preprint arXiv:1905.07769, 2019. ↑142

Huang, B., Zhao, B., and Song, Y.: Urban land-use mapping using a deep convolutional neural network with high spatial resolution multispectral remote sensing imagery, Remote Sens. Environ., 214, 73–86, 2018. ↑49

Huang, B., Li, X., Song, Z., and Yang, X.: Fl-ntk: A neural tangent kernel-based framework for federated learning analysis, in: International Conference on Machine Learning, pp. 4423–4434, PMLR, 2021. ↑116

Huang, G.-B., Zhu, Q.-Y., and Siew, C.-K.: Extreme learning machine: theory and applications, Neurocomputing, 70, 489–501, 2006. ↑148

Huang, H., Lin, L., Tong, R., Hu, H., Zhang, Q., Iwamoto, Y., Han, X., Chen, Y.-W., and Wu, J.: Unet 3+: A full-scale connected unet for medical image segmentation, in: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1055–1059, IEEE, 2020a. ↑45

Huang, J. and Yau, H.-T.: Dynamics of deep neural networks and neural tangent hierarchy, arXiv preprint arXiv:1909.08156, 2019. ↑142

Huang, K., Wang, Y., Tao, M., and Zhao, T.: Why Do Deep Residual Networks Generalize Better than Deep Feedforward Networks?—A Neural Tangent Kernel Perspective, Advances in neural information processing systems, 33, 2698–2709, 2020b. ↑117

Hubel, D. H. and Wiesel, T. N.: Receptive fields of single neurones in the cat's striate cortex, The Journal of physiology, 148, 574, 1959. ↑43

Hui, L. and Belkin, M.: Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks, arXiv preprint arXiv:2006.07322, 2020. ↑123

Isensee, F., Jaeger, P. F., Kohl, S. A., Petersen, J., and Maier-Hein, K. H.: nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation, Nature methods, 18, 203–211, 2021. ↑45

IUSS Working Group WRB: World reference base for soil resources, Rome, 2006. ↑51

Ivanovsky, L., Khryashchev, V., Pavlov, V., and Ostrovskaya, A.: Building detection on aerial images using U-NET neural networks, in: Conference of Open Innovation Association, FRUCT, pp. 116–122, 2019. ↑50

Jacot, A., Gabriel, F., and Hongler, C.: Neural tangent kernel: Convergence and generalization in neural networks, Advances in neural information processing systems, 31, 2018. ↑30, ↑32, ↑33, ↑116, ↑120, ↑141, ↑142

Javanmard, A., Mondelli, M., and Montanari, A.: Analysis of a two-layer neural network via displacement convexity, arXiv preprint arXiv:1901.01375, 2019. ↑142

Jha, A. H., Anand, S., Singh, M. K., and Veeravasarapu, V. S. R.: Disentangling Factors of Variation with Cycle-Consistent Variational Auto-Encoders, in: European Conference on Computer Vision, 2018. ↑90, ↑92, ↑96

Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D., and Bengio, S.: Fantastic Generalization Measures and Where to Find Them, in: International Conference on Learning Representations, 2020. ↑27

Jin, K. H., McCann, M. T., Froustey, E., and Unser, M.: Deep convolutional neural network for inverse problems in imaging, IEEE Transactions on Image Processing, 26, 4509–4522, 2017. ↑45

John, D. and Zhang, C.: An attention-based U-Net for detecting deforestation within satellite sensor imagery, International Journal of Applied Earth Observation and Geoinformation, 107, 102 685, 2022. ↑45

Johnson, R. A. and Wichern, D. W.: Applied multivariate statistical analysis, New Jersey, 405, 1992. ↑14

Jones, H., Springer, J. M., Kenyon, G. T., and Moore, J.: If You've Trained One You've Trained Them All: Inter-Architecture Similarity Increases With Robustness, in: The 38th Conference on Uncertainty in Artificial Intelligence, 2022. ↑152

Kanoh, R. and Sugiyama, M.: A Neural Tangent Kernel Perspective of Infinite Tree Ensembles, in: International Conference on Learning Representations, 2022. ↑116

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L.: Physics-informed machine learning, Nature Reviews Physics, 3, 422–440, 2021. ↑3

Kattenborn, T., Eichel, J., and Fassnacht, F. E.: Convolutional Neural Networks enable efficient, accurate and fine-grained segmentation of plant species and communities from high-resolution UAV imagery, Sci. Rep., 9, 17 656, 2019. ↑50, ↑64

Keller, S. M., Samarin, M., Wieser, M., and Roth, V.: Deep archetypal analysis, in: German Conference on Pattern Recognition, pp. 171–185, Springer, 2019. ↑42, ↑88, ↑90

Keller, S. M., Samarin, M., Torres, F. A., Wieser, M., and Roth, V.: Learning Extremal Representations with Deep Archetypal Analysis, International Journal of Computer Vision, 129, 805–820, 2021. ↑42, ↑88, ↑89, ↑90

Kessler, B.: An Empirical Study on the Finite and Infinite LeNet in the Context of the Neural Tangent Kernel, Master's thesis, University of Basel, 2021. ↑133, ↑134

Kim, H. and Mnih, A.: Disentangling by factorising, in: International Conference on Machine Learning, pp. 2649–2658, PMLR, 2018. ↑88, ↑90, ↑108

Kingma, D. P. and Ba, J.: Adam: A Method for Stochastic Optimization, in: International Conference on Learning Representations, 2015a. ↑24, ↑101

Kingma, D. P. and Ba, J. L.: Adam: A method for stochastic optimization, in: 3rd International Conference on Learning Representations, ICLR 2015, pp. 1–15, 2015b. ↑59

Kingma, D. P. and Welling, M.: Auto-Encoding Variational Bayes, in: 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings, edited by Bengio, Y. and LeCun, Y., 2014. ↑36, ↑39, ↑89, ↑91

Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M.: Semi-supervised learning with deep generative models, in: Advances in neural information processing systems, pp. 3581–3589, 2014. ↑89

Klys, J., Snell, J., and Zemel, R.: Learning Latent Subspaces in Variational Autoencoders, in: Advances in Neural Information Processing Systems, 2018. ↑88, ↑89, ↑90

Kohl, S., Romera-Paredes, B., Meyer, C., De Fauw, J., Ledsam, J. R., Maier-Hein, K., Eslami, S., Jimenez Rezende, D., and Ronneberger, O.: A probabilistic u-net for segmentation of ambiguous images, Advances in neural information processing systems, 31, 2018. ↑46, ↑145, ↑146

Kohl, S. A., Romera-Paredes, B., Maier-Hein, K. H., Rezende, D. J., Eslami, S., Kohli, P., Zisserman, A., and Ronneberger, O.: A hierarchical probabilistic u-net for modeling multi-scale ambiguities, arXiv preprint arXiv:1905.13077, 2019. ↑46, ↑145

Konz, N., Baenninger, D., Konz, M., Nearing, M., and Alewell, C.: Process identification of soil erosion in steep mountain regions, Hydrol. Earth Syst. Sci., 14, 675–686, 2010. ↑62

Konz, N., Prasuhn, V., and Alewell, C.: On the measurement of alpine soil erosion, Catena, 91, 63–71, 2012. ↑62

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G.: Similarity of neural network representations revisited, in: International Conference on Machine Learning, pp. 3519–3529, PMLR, 2019. ↑142, ↑152

Krizhevsky, A.: One weird trick for parallelizing convolutional neural networks, arXiv preprint arXiv:1404.5997, 2014. ↑26, ↑122, ↑159

Krizhevsky, A. and Hinton, G.: Learning multiple layers of features from tiny images, Citeseer, 2009. ↑123

Krizhevsky, A., Sutskever, I., and Hinton, G. E.: Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, pp. 1097–1105, 2012. ↑3, ↑43, ↑118

Kusner, M. J., Paige, B., and Hernández-Lobato, J. M.: Grammar Variational Autoencoder, in: International Conference on Machine Learning, 2017. ↑89

Lample, G., Zeghidour, N., Usunier, N., Bordes, A., Denoyer, L., and Ranzato, M.: Fader Networks: Manipulating Images by Sliding Attributes, in: Advances in Neural Information Processing Systems, 2017. ↑90

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D.: Backpropagation applied to handwritten zip code recognition, Neural computation, 1, 541–551, 1989. ↑43

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P.: Gradient-based learning applied to document recognition, Proceedings of the IEEE, 86, 2278–2324, 1998. ↑43, ↑118, ↑122, ↑123, ↑159

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations, in: Proceedings of the 26th annual international conference on machine learning, pp. 609–616, 2009. ↑119

Lee, J., Sohl-dickstein, J., Pennington, J., Novak, R., Schoenholz, S., and Bahri, Y.: Deep Neural Networks as Gaussian Processes, in: International Conference on Learning Representations, 2018. ↑28, ↑29, ↑116, ↑121

Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J.: Wide neural networks of any depth evolve as linear models under gradient descent, in: Advances in neural information processing systems, pp. 8570–8581, 2019. ↑30, ↑33, ↑34, ↑116, ↑117, ↑120, ↑141, ↑159

Lee, J., Schoenholz, S., Pennington, J., Adlam, B., Xiao, L., Novak, R., and Sohl-Dickstein, J.: Finite versus infinite neural networks: an empirical study, Advances in Neural Information Processing Systems, 33, 15 156–15 172, 2020. ↑121, ↑139, ↑141

Lin, Z., Thekumparampil, K., Fanti, G., and Oh, S.: Infogan-cr and model-centrality: Self-supervised model training and selection for disentangling gans, in: International Conference on Machine Learning, pp. 6127–6139, PMLR, 2020. ↑88, ↑90

Liu, B., Ding, M., Shaham, S., Rahayu, W., Farokhi, F., and Lin, Z.: When Machine Learning Meets Privacy: A Survey and Outlook, ACM Comput. Surv., 54, 2021. ↑1

Liu, H., Ong, Y.-S., Shen, X., and Cai, J.: When Gaussian process meets big data: A review of scalable GPs, IEEE transactions on neural networks and learning systems, 31, 4405–4423, 2020. ↑23

Locatello, F., Bauer, S., Lucic, M., Raetsch, G., Gelly, S., Schölkopf, B., and Bachem, O.: Challenging common assumptions in the unsupervised learning of disentangled representations, in: international conference on machine learning, pp. 4114–4124, PMLR, 2019. ↑88, ↑150

Long, J., Shelhamer, E., and Darrell, T.: Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3431–3440, 2015. ↑44

Louizos, C., Swersky, K., Li, Y., Welling, M., and Zemel, R. S.: The Variational Fair Autoencoder, in: 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings, edited by Bengio, Y. and LeCun, Y., 2016. ↑89

Louizos, C., Shalit, U., Mooij, J. M., Sontag, D., Zemel, R., and Welling, M.: Causal Effect Inference with Deep Latent-Variable Models, in: Advances in Neural Information Processing Systems 30, edited by Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., pp. 6446–6456, Curran Associates, Inc., 2017. ↑89

Lu, H., Ma, L., Fu, X., Liu, C., Wang, Z., Tang, M., and Li, N.: Landslides information extraction using Object-Oriented Image Analysis paradigm based on Deep Learning and Transfer Learning, Remote Sens., 12, 752, 2020. ↑64

Ma, L., Liu, Y., Zhang, X., Ye, Y., Yin, G., and Johnson, B. A.: Deep learning in remote sensing applications: A meta-analysis and review, ISPRS J. Photogramm. Remote Sens., 152, 166–177, 2019. ↑49

Marcinkevičs, R. and Vogt, J. E.: Interpretability and explainability: A machine learning zoo mini-tour, arXiv preprint arXiv:2012.01805, 2020. ↑3

Martha, T. R., Kerle, N., van Westen, C. J., Jetten, V., and Vinod Kumar, K.: Object-oriented analysis of multi-temporal panchromatic images for creation of historical landslide inventories, ISPRS J. Photogramm. Remote Sens., 67, 105–119, 2012. ↑49

Matthey, L., Higgins, I., Hassabis, D., and Lerchner, A.: dSprites: Disentanglement testing Sprites dataset, https://github.com/deepmind/dsprites-dataset/, 2017. ↑107

Mboga, N., Georganos, S., Grippa, T., Lennert, M., Vanhuysse, S., and Wolff, E.: Fully convolutional networks and geographic object-based image analysis for the classification of VHR imagery, Remote Sens., 11, 597, 2019. ↑50, ↑69

Mei, S., Misiakiewicz, T., and Montanari, A.: Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit, in: Proceedings of the Thirty-Second Conference on Learning Theory, edited by Beygelzimer, A. and Hsu, D., vol. 99, pp. 2388–2464, PMLR, 2019. ↑142

Meusburger, K. and Alewell, C.: Impacts of anthropogenic and environmental factors on the occurrence of shallow landslides in an alpine catchment (Urseren Valley, Switzerland), Nat. Hazards Earth Syst. Sci., 8, 509–520, 2008. ↑9, ↑48, ↑51, ↑52

Meusburger, K. and Alewell, C.: Soil Erosion in the Alps, Federal Office for the Environment FOEN, p. 118, 2014. ↑10, ↑62

Meusburger, K., Bänninger, D., and Alewell, C.: Estimating vegetation parameter for soil erosion assessment in an alpine catchment by means of QuickBird imagery, Int. J. Appl. Earth Obs. Geoinf., 12, 201–207, 2010. ↑49

Meusburger, K., Steel, A., Panagos, P., Montanarella, L., and Alewell, C.: Spatial and temporal variability of rainfall erosivity factor for Switzerland, Hydrol. Earth Syst. Sci., 16, 167–177, 2012. ↑49

Murphy, K. P.: Machine learning: a probabilistic perspective, MIT press, 2012. ↑13, ↑15, ↑18, ↑22

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I.: Deep double descent: Where bigger models and more data hurt, Journal of Statistical Mechanics: Theory and Experiment, 2021, 124 003, 2021. ↑26, ↑27

Neal, B., Mittal, S., Baratin, A., Tantia, V., Scicluna, M., Lacoste-Julien, S., and Mitliagkas, I.: A modern take on the bias-variance tradeoff in neural networks, arXiv preprint arXiv:1810.08591, 2018. ↑27

Neal, R. M.: Bayesian learning for neural networks, vol. 118, Lecture Notes in Statistics, 1996. ↑28, ↑116

Nearing, M., Pruski, F., and O'Neal, M.: Expected climate change impacts on soil erosion rates: A review, J. Soil Water Conserv., 59, 43–50, 2004. ↑9, ↑48

Nesterov, V., Wieser, M., and Roth, V.: 3DMolNet: A Generative Network for Molecular Structures, 2020. ↑89

Nesterov, V., Torres, F. A., Nagy-Huber, M., Samarin, M., and Roth, V.: Learning Invariances with Generalised Input-Convex Neural Networks, arXiv preprint arXiv:2204.07009, 2022. ↑151

Neyshabur, B., Li, Z., Bhojanapalli, S., LeCun, Y., and Srebro, N.: The role of over-parametrization in generalization of neural networks, in: International Conference on Learning Representations, 2019. ↑26

Nguyen, P.-M.: Mean field limit of the learning dynamics of multilayer neural networks, arXiv preprint arXiv:1902.02880, 2019. ↑142

Novak, R., Xiao, L., Lee, J., Bahri, Y., Yang, G., Hron, J., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J.: Bayesian deep convolutional networks with many channels are gaussian processes, arXiv preprint arXiv:1810.05148, 2018. ↑28

Novak, R., Xiao, L., Hron, J., Lee, J., Alemi, A. A., Sohl-Dickstein, J., and Schoenholz, S. S.: Neural Tangents: Fast and Easy Infinite Neural Networks in Python, in: 8th International Conference on Learning Representations, ICLR, Addis Ababa, Ethiopia, April 26-30, 2020. ↑123

Oktay, O., Schlemper, J., Folgoc, L. L., Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N. Y., Kainz, B., Glocker, B., and Rueckert, D.: Attention U-Net: Learning Where to Look for the Pancreas, in: Medical Imaging with Deep Learning, 2018. ↑45

Ortiz-Jiménez, G., Moosavi-Dezfooli, S.-M., and Frossard, P.: What can linearized neural networks actually say about generalization?, Advances in Neural Information Processing Systems, 34, 2021. ↑121

Pan, X., Zhao, J., and Xu, J.: An object-based and heterogeneous segment filter convolutional neural network for high-resolution remote sensing image classification, Int. J. Remote Sens., 40, 5892–5916, 2019. ↑71

Parbhoo, S., Wieser, M., Roth, V., and Doshi-Velez, F.: Transfer Learning from Well-Curated to Less-Resourced Populations with HIV, in: Proceedings of the 5th Machine Learning for Healthcare Conference, 2020a. ↑90

Parbhoo, S., Wieser, M., Wieczorek, A., and Roth, V.: Information Bottleneck for Estimating Treatment Effects with Systematically Missing Covariates, Entropy, 22, 389, 2020b. ↑90

Park, J. and Sandberg, I. W.: Universal Approximation Using Radial-Basis-Function Networks, Neural Computation, 3, 246–257, 1991. ↑148

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S.: PyTorch: An Imperative Style, High-Performance Deep Learning Library, in: Advances in Neural Information Processing Systems 32, edited by Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., pp. 8024–8035, Curran Associates, Inc., 2019. ↑122

Prakash, N., Manconi, A., and Loew, S.: Mapping landslides on EO data: Performance of deep learning models vs. Traditional machine learning models, Remote Sens., 12, 2020. ↑71

Prasuhn, V., Liniger, H., Gisler, S., Herweg, K., Candinas, A., and Clément, J. P.: A high-resolution soil erosion risk map of Switzerland as strategic policy support system, Land Use Policy, 32, 281–291, 2013. ↑49

Rahimi, A. and Recht, B.: Random features for large-scale kernel machines, Advances in neural information processing systems, 20, 2007. ↑117, ↑148

Rahimi, A. and Recht, B.: Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning, Advances in neural information processing systems, 21, 2008. ↑148

Ramakrishnan, R., Dral, P. O., Rupp, M., and Von Lilienfeld, O. A.: Quantum chemistry structures and properties of 134 kilo molecules, Scientific data, 1, 1–7, 2014. ↑105

Raman, S., Fuchs, T. J., Wild, P. J., Dahl, E., and Roth, V.: The Bayesian Group-Lasso for Analyzing Contingency Tables, in: Proceedings of the 26th Annual International Conference on Machine Learning, 2009. ↑90

Rasmussen, C. E. and Williams, C. K. I.: Gaussian processes for machine learning, vol. 2, MIT press Cambridge, MA, 2006. ↑20, ↑22

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A.: You only look once: Unified, real-time object detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779–788, 2016. ↑43

Rey, M., Roth, V., and Fuchs, T.: Sparse meta-Gaussian information bottleneck, in: International Conference on Machine Learning, pp. 910–918, PMLR, 2014. ↑41, ↑88, ↑90, ↑94

Rezende, D. and Mohamed, S.: Variational inference with normalizing flows, in: International conference on machine learning, pp. 1530–1538, PMLR, 2015. ↑36

Rezende, D. J., Mohamed, S., and Wierstra, D.: Stochastic backpropagation and approximate inference in deep generative models, in: International conference on machine learning, pp. 1278–1286, PMLR, 2014. ↑36, ↑89, ↑91

Robert, T., Thome, N., and Cord, M.: DualDis: Dual-Branch Disentangling with Adversarial Learning, 2019. ↑88

Ronneberger, O., Fischer, P., and Brox, T.: U-net: Convolutional networks for biomedical image segmentation, in: Medical Image Computing and Computer-assisted intervention - MICCAI 2015, edited by Navab, N., Hornegger, J., Wells, W., and Frangi, A., vol. 9351, pp. 234–241, 2015. ↑44, ↑49, ↑56

Rotskoff, G. M., Jelassi, S., Bruna, J., and Vanden-Eijnden, E.: Global convergence of neuron birth-death dynamics, in: 36th International Conference on Machine Learning, pp. 9689–9698, 2019. ↑142

Roy, O. and Vetterli, M.: The effective rank: A measure of effective dimensionality, in: 15th European Signal Processing Conference, pp. 606–610, IEEE, 2007. ↑129, ↑161

Rumelhart, D. E., Hinton, G. E., and Williams, R. J.: Learning representations by back-propagating errors, nature, 323, 533–536, 1986. ↑24

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge, International Journal of Computer Vision (IJCV), 115, 211–252, 2015. ↑124

Ruthotto, L. and Haber, E.: An introduction to deep generative modeling, GAMM-Mitteilungen, 44, e202100 008, 2021. ↑36

Samarin, M., Zweifel, L., Roth, V., and Alewell, C.: Identifying soil erosion processes in alpine grasslands on aerial imagery with a u-net convolutional neural network, Remote Sensing, 12, 4149, 2020. ↑5, ↑47

Samarin, M., Nesterov, V., Wieser, M., Wieczorek, A., Parbhoo, S., and Roth, V.: Learning Conditional Invariance Through Cycle Consistency, in: DAGM German Conference on Pattern Recognition, pp. 376–391, Springer, 2021. ↑5, ↑87

Samarin, M., Roth, V., and Belius, D.: Feature Learning and Random Features in Standard Finite-Width Convolutional Neural Networks: An Empirical Study, in: The 38th Conference on Uncertainty in Artificial Intelligence, 2022. ↑5, ↑115

Scheurer, K., Alewell, C., Bänninger, D., and Burkhardt-holm, P.: Climate and land-use changes affecting river sediment and brown trout in alpine countries - a review, Environ. Sci. Pollut. Res., 16, 232–242, 2009. ↑48

Schmidt, S., Alewell, C., and Meusburger, K.: Mapping spatio-temporal dynamics of the cover and management factor (C-factor) for grasslands in Switzerland, Remote Sens. Environ., 211, 89–104, 2018. ↑49

Schmidt, S., Alewell, C., and Meusburger, K.: Monthly RUSLE soil erosion risk of Swiss grasslands, J. Maps, 15, 247–256, 2019a. ↑49

Schmidt, S., Tresch, S., and Meusburger, K.: Modification of the RUSLE slope length and steepness factor (LS-factor) based on rainfall experiments at steep alpine grasslands, MethodsX, 6, 219–229, 2019b. ↑49

Schölkopf, B. and Smola, A. J.: Learning with kernels: support vector machines, regularization, optimization, and beyond, MIT press, 2002. ↑13, ↑19, ↑20

Seleznova, M. and Kutyniok, G.: Analyzing Finite Neural Networks: Can We Trust Neural Tangent Kernel Theory?, 2022. ↑116, ↑121

Shruthi, R. B. V., Kerle, N., and Jetten, V.: Object-based gully feature extraction using high spatial resolution imagery, Geomorphology, 134, 260–268, 2011. ↑49

Simon, N., Friedman, J., Hastie, T., and Tibshirani, R.: A sparse-group lasso, Journal of Computational and Graphical Statistics, 2013. ↑90

Simonyan, K. and Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition, in: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, edited by Bengio, Y. and LeCun, Y., 2015. ↑120

Socher, R., Lin, C. C., Manning, C., and Ng, A. Y.: Parsing natural scenes and natural language with recursive neural networks, in: Proceedings of the 28th international conference on machine learning (ICML-11), pp. 129–136, 2011. ↑3

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics, in: International Conference on Machine Learning, pp. 2256–2265, PMLR, 2015. ↑36

Song, J. and Ermon, S.: Understanding the Limitations of Variational Mutual Information Estimators, in: International Conference on Learning Representations, 2020. ↑96

Song, Y. and Ermon, S.: Generative modeling by estimating gradients of the data distribution, Advances in Neural Information Processing Systems, 32, 2019. ↑36

Spigler, S., Geiger, M., d'Ascoli, S., Sagun, L., Biroli, G., and Wyart, M.: A jamming transition from under-to over-parametrization affects generalization in deep learning, Journal of Physics A: Mathematical and Theoretical, 52, 474 001, 2019. ↑26

Sun, C., Shrivastava, A., Singh, S., and Gupta, A.: Revisiting unreasonable effectiveness of data in deep learning era, in: Proceedings of the IEEE international conference on computer vision, pp. 843–852, 2017. ↑152

Swisstopo: Swissimage. Das digitale Farborthophotomosaik der Schweiz., 2010. ↑52

Swisstopo: SwissALTI3D. Das hoch aufgelöste Terrainmodell der Schweiz., 2014. ↑53

Tasser, E., Mader, M., and Tappeiner, U.: Effects of land use in alpine grasslands on the probability of landslides, Basic Appl. Ecol., 4, 271–280, 2003. ↑48

Tibshirani, R.: Regression Shrinkage and Selection via the Lasso, Journal of the Royal Statistical Society (Series B), 1996. ↑16, ↑88, ↑90

Tishby, N., Pereira, F. C., and Bialek, W.: The information bottleneck method, in: Allerton Conference on Communication, Control and Computing, 1999. ↑39, ↑41, ↑88, ↑89, ↑91

Torresani, L., Wu, J., Masin, R., Penasa, M., and Tarolli, P.: Estimating soil degradation in montane grasslands of North-eastern Italian Alps (Italy), Heliyon, 5, e01 825, 2019. ↑48

Tunyasuvunakool, K., Adler, J., Wu, Z., Green, T., Zielinski, M., Žídek, A., Bridgland, A., Cowie, A., Meyer, C., Laydon, A., Velankar, S., Kleywegt, G. J., Bateman, A., Evans, R., Pritzel, A., Figurnov, M., Ronneberger, O., Bates, R., Kohl, S. A. A., Potapenko, A., Ballard, A. J., Romera-Paredes, B., Nikolov, S., Jain, R., Clancy, E., Reiman, D., Petersen, S., Senior, A. W., Kavukcuoglu, K., Birney, E., Kohli, P., Jumper, J., and Hassabis, D.: Highly accurate protein structure prediction for the human proteome, Nature, 596, 590–596, 2021. ↑3

Valle-Perez, G., Camargo, C. Q., and Louis, A. A.: Deep learning generalizes because the parameter-function map is biased towards simple functions, in: International Conference on Learning Representations, 2019. ↑27, ↑28

Vapnik, V.: The nature of statistical learning theory, Springer science & business media, 1999. ↑26

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I.: Attention is all you need, Advances in neural information processing systems, 30, 2017. ↑43, ↑45

Wang, B., Zhang, Z., Wang, X., Zhao, X., Yi, L., and Hu, S.: Object-based mapping of gullies using optical images: A case study in the black soil region, Northeast of China, Remote Sens., 12, 487, 2020. ↑49

Wei, C., Lee, J. D., Liu, Q., and Ma, T.: Regularization Matters: Generalization and Optimization of Neural Nets v.s. their Induced Kernel, 2020. ↑121

Wieczorek, A. and Roth, V.: Causal compression, arXiv preprint arXiv:1611.00261, 2016. ↑88

Wieczorek, A. and Roth, V.: On the difference between the Information Bottleneck and the Deep Information Bottleneck, Entropy, 22, 131, 2020. ↑42

Wieczorek, A., Wieser, M., Murezzan, D., and Roth, V.: Learning Sparse Latent Representations with the Deep Copula Information Bottleneck, in: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings, OpenReview.net, 2018. ↑88, ↑89, ↑90, ↑94

Wiegand, C. and Geitner, C.: Shallow erosion in grassland areas in the Alps. What we know and what we need to investigate further, in: Challenges for Mountain Regions: Tackling Complexity, p. 76–83, 2010. ↑48

Wiegand, C., Rutzinger, M., Heinrich, K., and Geitner, C.: Automated extraction of shallow erosion areas based on multi-temporal ortho-imagery, Remote Sens., 5, 2292–2307, 2013. ↑49

Wieser, M.: Learning Invariant Representations for Deep Latent Variable Models, Ph.D. thesis, University_of_Basel, 2020. ↑90, ↑96

Wieser, M., Parbhoo, S., Wieczorek, A., and Roth, V.: Inverse Learning of Symmetries, in: Advances in Neural Information Processing Systems, 2020. ↑42, ↑88, ↑89, ↑90, ↑96, ↑99

Wilkinson, J., Arnold, K. F., Murray, E. J., van Smeden, M., Carr, K., Sippy, R., de Kamps, M., Beam, A., Konigorski, S., Lippert, C., Gilthorpe, M., and Tennant, P.: Time to reality check the promises of machine learning-powered precision medicine, The Lancet Digital Health, 2, e677–e680, 2020. ↑1

Williams, C.: Computing with infinite networks, Advances in neural information processing systems, 9, 1996. ↑28, ↑116

Wu, M., Hughes, M. C., Parbhoo, S., Zazzi, M., Roth, V., and Doshi-Velez, F.: Beyond Sparsity: Tree Regularization of Deep Models for Interpretability, 2017. ↑88

Wulamu, A., Shi, Z., Zhang, D., and He, Z.: Multiscale Road Extraction in Remote Sensing Images, Comput. Intel. Neurosc., 2019, 1–9, 2019. ↑50

Wyner, A. J., Olson, M., Bleich, J., and Mease, D.: Explaining the success of adaboost and random forests as interpolating classifiers, The Journal of Machine Learning Research, 18, 1558–1590, 2017. ↑26

Wyss, R.: Die Urseren-Zone - Lithostratigraphie und Tektonik, Eclogae Geol. Hel., 79, 731–767, 1986. ↑51

Xiao, L., Pennington, J., and Schoenholz, S.: Disentangling trainability and generalization in deep neural networks, in: International Conference on Machine Learning, pp. 10 462–10 472, PMLR, 2020. ↑116

Xu, Y., Wu, L., Xie, Z., and Chen, Z.: Building extraction in very high resolution remote sensing imagery using deep learning and guided filters, Remote Sens., 10, 144, 2018. ↑50

Yang, G.: Wide feedforward or recurrent neural networks of any architecture are gaussian processes, Advances in Neural Information Processing Systems, 32, 2019a. ↑28

Yang, G.: Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation, arXiv preprint arXiv:1902.04760, 2019b. ↑117

Yang, G.: Tensor programs ii: Neural tangent kernel for any architecture, arXiv preprint arXiv:2006.14548, 2020. ↑117

Yang, G. and Hu, E. J.: Tensor programs iv: Feature learning in infinite-width neural networks, in: International Conference on Machine Learning, pp. 11 727–11 737, PMLR, 2021. ↑121

Yang, J., Guo, J., Yue, H., Liu, Z., Hu, H., and Li, K.: CDnet: CNN-based cloud detection for remote sensing imagery, IEEE Trans. Geosci. Remote Sens., 57, 6195–6211, 2019. ↑50

Ye, J. C.: Geometry of Deep Learning: A Signal Processing Perspective, vol. 37, Springer Nature, 2022. ↑13

Yi, Y., Zhang, Z., Zhang, W., Zhang, C., Li, W., and Zhao, T.: Semantic segmentation of urban buildings from VHR remote sensing imagery using a deep convolutional neural network, Remote Sens., 11, 1774, 2019. ↑50

Yuan, M., Liu, Z., and Wang, F.: Using the wide-range attention u-net for road segmentation, Remote Sens. Lett., 10, 506–515, 2019. ↑50

Yuan, Q., Shen, H., Li, T., Li, Z., Li, S., Jiang, Y., Xu, H., Tan, W., Yang, Q., Wang, J., Gao, J., and Zhang, L.: Deep learning in environmental remote sensing: Achievements and challenges, Remote Sens. Environ., 241, 111 716, 2020. ↑49

Zagoruyko, S. and Komodakis, N.: Wide residual networks, arXiv preprint arXiv:1605.07146, 2016. ↑120

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O.: Understanding deep learning requires rethinking generalization, in: International Conference on Learning Representations, 2017. ↑139

Zhang, C., Sargent, I., Pan, X., Li, H., Gardiner, A., Hare, J., and Atkinson, P. M.: An object-based convolutional neural network (OCNN) for urban land use classification, Remote Sensing of Environment, 216, 57–70, 2018a. ↑64

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O.: Understanding deep learning (still) requires rethinking generalization, Communications of the ACM, 64, 107–115, 2021. ↑139

Zhang, S., Bamakan, S. M. H., Qu, Q., and Li, S.: Learning for Personalized Medicine: A Comprehensive Review From a Deep Learning Perspective, IEEE Reviews in Biomedical Engineering, 12, 194–208, 2019. ↑1

Zhang, Z., Geiger, J., Pohjalainen, J., Mousa, A. E.-D., Jin, W., and Schuller, B.: Deep learning for environmentally robust speech recognition: An overview of recent developments, ACM Transactions on Intelligent Systems and Technology (TIST), 9, 1–28, 2018b. ↑3

Zhang, Z., Liu, Q., and Wang, Y.: Road Extraction by Deep Residual U-Net, IEEE Geosci. Remote Sens. Lett., 15, 749–753, 2018c. ↑50, ↑64, ↑69

Zhou, Z., Rahman Siddiquee, M. M., Tajbakhsh, N., and Liang, J.: Unet++: A nested u-net architecture for medical image segmentation, in: Deep learning in medical image analysis and multimodal learning for clinical decision support, pp. 3–11, Springer, 2018. ↑45

Zhu, J., Park, T., Isola, P., and Efros, A. A.: Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks, in: International Conference on Computer Vision, 2017. ↑92

Zou, D., Cao, Y., Zhou, D., and Gu, Q.: Gradient descent optimizes over-parameterized deep ReLU networks, Machine Learning, 109, 467–492, 2020. ↑119

Zweifel, L.: Identifying Soil Erosion Processes in the Alps using Machine Learning Techniques, Ph.D. thesis, University of Basel, 2021. ↑5, ↑10, ↑47

Zweifel, L., Meusburger, K., and Alewell, C.: Spatio-temporal pattern of soil degradation in a Swiss Alpine grassland catchment, Remote Sens. Environ., 235, 111 441, 2019. ↑9, ↑48, ↑49, ↑52, ↑53, ↑56, ↑59, ↑62, ↑63, ↑67

Zweifel, L., Samarin, M., Meusburger, K., and Alewell, C.: Investigating causal factors of shallow landslides in grassland regions of Switzerland, Natural Hazards and Earth System Sciences, 21, 3421–3437, 2021. ↑72