THESIS FOR THE DEGREE OF LICENTIATE OF TECHNOLOGY

Characterizing Piecewise Linear Neural Networks

Theoretical and Applied Perspectives

Anton Johansson

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Division of Applied Mathematics and Statistics
Department of Mathematical Sciences
Chalmers University of Technology and University of Gothenburg
Göteborg, Sweden 2022

Characterizing Piecewise Linear Neural Networks: Theoretical and Applied Perspectives
Anton Johansson
Göteborg 2022

Division of Applied Mathematics and Statistics
Department of Mathematical Sciences
Chalmers University of Technology and University of Gothenburg
SE-412 96 Göteborg
Sweden
Telephone +46 (0)31 772 1000

Typeset with LaTeX
Printed by Chalmers Reproservice
Göteborg, Sweden 2022

Characterizing Piecewise Linear Neural Networks
Theoretical and Applied Perspectives

Anton Johansson

Division of Applied Mathematics and Statistics
Department of Mathematical Sciences
Chalmers University of Technology and University of Gothenburg

Abstract

Neural networks utilizing piecewise linear transformations between layers have in many regards become the default network type to use across a wide range of applications. Their superior training dynamics and generalization performance irrespective of the nature of the problem has resulted in these networks achieving state of the art results on a diverse set of tasks.

Even though the efficacy of these networks have been established, there is a poor understanding of their intrinsic behaviour and properties. Little is known regarding how these functions evolve during training, how they behave at initialization and how all of this is related to the architecture of the network. Exploring and detailing these properties is not only of theoretical interest, it can also aid in developing new schemes and algorithms to further improve the performance of the networks.

In this thesis we thus seek to further explore and characterize these properties. We theoretically prove how the local properties of piecewise linear networks vary at initialization and explore empirically how more complex properties behave during training. We use these results to reason about which intrinsic properties are associated with the generalization performance and develop new regularization schemes. We further substantiate the empirical success of piecewise linear networks by showcasing how their application can solve two tasks relevant to the safety and effectiveness of processes related to the automotive industry.

Keywords: machine learning, neural network, piecewise linear, automotive applications.

List of publications

This thesis is based on the work represented by the following papers:

I. D. Parthasarathy and **A. Johansson** "Does the dataset meet your expectations? Explaining sample representation in image data"
In proceedings of BNAIC/Benelearn, 2020.

II. **A. Johansson**, N. Engsner, C. Strannegård, P. Mostad "Slope and generalization properties of neural networks"
arXiv preprint arXiv:2107.01473.

III. D. Parthasarathy and **A. Johansson** "SilGAN: Generating driving maneuvers for scenario-based software-in-the-loop testing"
In proceedings of the third IEEE International Conference On Artificial Intelligence Testing 2021.

IV. **A. Johansson**, N. Engsner, C. Strannegård, P. Mostad "Improved Spectral Norm Regularization for Neural Networks"
Submitted.

Additional papers not included in this thesis:

V. D. Chatterjee, S. Ahlinder, **A. Johansson** "DataCleaningTool: An open-source MATLAB app for cooperative data cleaning"
Submitted

Author contributions

I. Conducted auxiliary experiments, had regular meetings with the main author during the latter part of the project, jointly developed the overlap index and assisted with writing the paper.

II. Formulated some of the hypotheses and programmed all experiments. Wrote large parts of the paper.

III. Conducted auxiliary experiments, had frequent meetings with the main author to discuss the development of the method, proposed the expansion stage of the model and assisted with writing the paper.

IV. Devised the regularization method, conducted the experiments and wrote the paper.

Acknowledgements

Thank you Petter for the insights and support that you have given me throughout these years. Performing research can at times be taxing but I felt that our discussions helped me and made it enjoyable even at times when there was no major significant output.

Thank you Niklas and Claes for all of the meetings, suggestions and general discussions regarding the projects and related fields.

Thank you Dhas for giving me the opportunity to experience a more applied aspect of research.

Thank you to the people at the math department for making it a great place to work.

Thank you Mutsuki for your patience and love.

Contents

1 Introduction

As the empirical success of deep learning methods is becoming progressively more evident, forming crucial components for technologies such as speech recognition [Roger et al. (2020)], object detection [Redmon et al. (2016)] and natural language processing [Otter et al. (2021)], there is an ever growing need to understand the inner workings of the method theoretically. For example, being able to prove for which tasks or scenarios a network is prone to yield erroneous results can not only enable engineers to quickly test and develop new prototypes, it also forms a crucial aspect of deploying deep learning in safety critical scenarios.

While it is human to strive for a full theory in all generality, such a theory of of deep learning is likely to be an insurmountable task for years to come. It is thus more fruitful to focus our efforts on the behaviour of the method in a restricted setting, obtaining partial theories and explanations that can shed light on the behaviour of deep learning in a limited context.

A restricted setting that can be simpler to analyze is that of the neural networks for which the mapping at inference time between the input and output space is piecewise linear. Such a mapping arises when all non-linear transformations between the layers in the network are given by piecewise linear functions, e.g., by utilizing ReLU activation functions [Nair and Hinton (2010)], max-pooling [Gholamalinezhad and Khosravi (2020)] or certain normalization layers [Ioffe and Szegedy (2015)].

These networks have not only achieved state of the art results on several tasks [He et al. (2016); Redmon et al. (2016); Szegedy et al. (2015)], they can additionally be decomposed into simpler constituent parts where the function is completely described by an affine relation. A global understanding of these networks and their behaviour could thus be facilitated by understanding the characteristics of these local constituent parts and their behavior. Such a characterization of the local properties of the piecewise linear networks will establish

a firm foundation from which one can reason about the method, paving the way for a comprehensive understanding of deep learning in this restricted regime.

We thus undertake the task of further facilitating this characterization. In this endeavor we explore and detail the properties of these networks, both by theoretically proving how they behave at initialization and also by empirically demonstrating their behaviour during training. We use our results to reason about the generalizability of deep learning and develop new regularization schemes. Additionally, we further add on to the empirical success of deep learning by demonstrating how these networks can allow us to approach and solve novel tasks where classical machine learning methods fail, here demonstrated as the solution to the problem of detecting possible failure modes for models trained on immense data sets and for the problem of generating realistic automotive test-stimuli from simple specifications.

The structure of the thesis is as follows. In Section 1.1 we first introduce the preliminaries for understanding the included papers, introducing what neural networks are, how they can be used in machine learning and additionally the *geometry* that arises from working with piecewise linear networks. In Section 2 we showcase some minor results regarding piecewise linear networks, both empirical results for how the functions behave during training but also theoretical results regarding their behaviour at initialization. Finally, in Section 3 and onwards we introduce the main papers of this thesis.

1.1 Preliminaries

1.1.1 Neural networks

Let us start with defining what a neural network is. Working with the same definition as in **Paper II** [Johansson et al. (2021)], a neural network $f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_c}$ will for us consist of

- a sequence of positive integers $n_0, n_1, ..., n_n = n_c$, where $n_1, \ldots, n_{n-1}$ denote the width of the hidden layers,
- for $i = 1, ..., n$, an $(n_i \times n_{i-1})$-dimensional matrix W_i and a vector b_i of length n_i, and
- a continuous activation function $g : \mathbb{R} \to \mathbb{R}$ applied separately to each dimension.

We define $f^0(x) = x$ and for $i = 1, \dots, n-1$ a continuous map $f^i : \mathbb{R}^{n_0} \to \mathbb{R}^{n_i}$ by setting

$$f^i(x) = g(W_i f^{i-1}(x) + b_i)$$

while we set $f(x) = f^n(x) = W_n f^{n-1}(x) + b_n$. A commonplace terminology that we will follow is that the output of *neuron* j of layer $l < n$ is given by the j:th component of $f^l(x)$.

If not otherwise mentioned, we will refer to the sets of all matrices W_i and vectors b_i of the network as the *weights* and *biases* of the network. These sets will be denoted as $\mathcal{W}$ and $\mathcal{B}$ respectively and the union of the sets $\theta := \mathcal{W} \cup \mathcal{B}$ will be referred to as the *parameters* of the network. We will at times write the network function as $f(\cdot; \theta)$ to make the dependence of the network on the parameters explicit.

If we momentarily restrict ourselves to regression for simplicity, then the supervised machine learning task is to learn parameters θ such that the network learns to associate feature vectors $x \in \mathbb{R}^{n_0}$ with a corresponding value-vector $y \in \mathbb{R}^{n_c}$. This association is achieved by first sampling a training set $\mathcal{D}_t$ of feature-value pairs from some distribution P, $\mathcal{D}_t := \{(x_i, y_i)\}_{i=1}^N$, $(x_i, y_i) \sim P \, \forall i$, and then employing a suitable loss function $l : \mathbb{R}^{n_c} \times \mathbb{R}^{n_c} \to \mathbb{R}$ to minimize the discrepancy between the predicted value $f(x_i)$ and associated value-vector y_i, where $(x_i, y_i) \in \mathcal{D}_t$. For example, the archetypal loss function for regression can be argued to be the l_p-loss given as $l_p(x_1, x_2) = ||x_1 - x_2||_p^p$, where $|| \cdot ||_p$ denotes the p-norm, most common choices being $p = 1$ or 2. For this loss we may formulate the learning task as

$$\min_\theta \underbrace{\sum_{(x_i, y_i) \in \mathcal{D}_t} l_p(f(x_i; \theta), y_i)}_{:= l(\theta; \mathcal{D}_t)} \tag{1.1}$$

which can be minimized through some variant of gradient descent. The typical variant is to work with some form of stochastic gradient descent (SGD) where the gradient is calculated only for a *batch*, meaning a subset of the training data, at each iteration.

1.1.2 Generalization

For a wide variety of tasks the minimization of (1.1) can be relatively easy given that the network has sufficiently many parameters. However, this minimization does not mean that the network *generalizes*, meaning that the network might have learned parameters θ that only rely on spurious associations that are only

present in $\mathcal{D}_t$ but that will not minimize $l(\theta; \mathcal{D}_v)$ for $\mathcal{D}_v \cap \mathcal{D}_t = \emptyset$, $\mathcal{D}_v \ni (x_i, y_i) \sim P$.

Many techniques exist to guide the minimization descent such that the network does not learn spurious associations and instead generalizes, e.g.,

- Regularization where a penalty term $R(\theta)$ is added to the minimization objective which penalizes solutions that are more 'complex'. For example, weight decay which penalizes the magnitude of the weights squared.
- Small batch sizes which adds a stochastic element to the descent.
- Early stopping where a separate validation set $\mathcal{D}_v$ is obtained, with $\mathcal{D}_v \cap \mathcal{D}_t = \emptyset$, $\mathcal{D}_v \ni (x_i, y_i) \sim P$, and at each iteration of gradient descent the two terms $l(\theta; \mathcal{D}_v)$ and $l(\theta; \mathcal{D}_t)$ are calculated to gauge if the current parameters θ are generalizing or not. At the iteration where $l(\theta; \mathcal{D}_v)$ increases but $l(\theta; \mathcal{D}_t)$ decreases one stops the gradient descent.

1.1.3 Generative adversarial network

One of the key examples demonstrating the power of neural networks and one we rely on in **Paper III** is that of the Generative Adversarial Network (GAN) [Goodfellow et al. (2014); Schmidhuber (2020)]. The GAN method has shown the capacity to be able to solve exceptionally complicated tasks which previous methods struggled with, e.g., generating art following a certain style, creating realistic hand-writing and speech synthesis [Kong et al. (2020); Shahriar (2021)].

While GANs can be used for supervised tasks, it is most straightforward to introduce the method in an unsupervised setting. We thus have a training set without associated labels, meaning $\mathcal{D}_t = \{x_i\}_{i=1}^N$, and we seek to approximate the distribution P that generated the training data. In the most simple approach to this problem we work with two neural networks, a generator G and a discriminator D. The generator attempts to learn to transform samples z from a prior distribution $p(z)$ such that they follow the distribution of interest P, i.e., $G(z) \sim P$. Simultaneously, the discriminator strives to learn to differentiate between the generated samples $G(z)$ and the true samples $x \in \mathcal{D}_t$ by outputting a probability which indicates how likely the input is to follow the distribution P. The full learning objective for this competition can be formulated as a minmax game as

$$\min_G \max_D V(G, D) = \mathbb{E}_{x \sim P}[\log D(x)] + \mathbb{E}_{z \sim p(z)}[1 - D(G(z))], \tag{1.2}$$

which can be approximately solved by sequentially training the generator and discriminator, ultimately allowing us to approximate complex distributions by simply specifying networks of sufficiently high complexity and running SGD on the loss objective (1.2).

While this basic scheme can allow us to tackle tasks which are intractable with classical machine learning algorithms, several improvements have been made to extend and improve upon the GAN scheme. For example, one can extend the scheme to model conditional distributions $p(x|y)$ by supplying different conditions y to both the generator and discriminator, altering the loss function as

$$\min_G \max_D V(G, D) = \mathbb{E}_{x\sim p(x|y)}[\log D(x, y)] + \mathbb{E}_{z\sim p(z)}[1 - D(G(z, y))]. \quad (1.3)$$

With such a scheme one can, e.g., develop models that transform sketches to realistic photos [Isola et al. (2017)] and generate true to life images from text [Reed et al. (2016)].

1.1.4 Geometric view of piecewise linear networks

In **Paper II** we utilize a geometric perspective of piecewise linear neural networks to characterize their behaviour. With this geometric perspective we decompose the input space into smaller constituent regions where the function is completely described by an affine relation, see Figure 1.1.

With this perspective we can analyze the behaviour of the network by analyzing the behaviour of the local affine transformations and associated regions, obtaining global conclusions from this local view. For example, if certain network architectures predispose adjacent regions to be extremely dissimilar then it is likely that such a network will not be able to model the smooth variations present in many physical phenomena, making it likely that the network will not generalize on a wide variety of tasks. Another benefit of this perspective is that understanding and posing constraints on the network can be simpler if done in terms of the local characteristics of the regions and the affine transformations, e.g., a constraint on the network to be robust against perturbations can be understood as a constraint on the singular values of the affine transformations.

While piecewise linear networks can be obtained by a several different piecewise linear transformations between the layers, for simplicity of analysis in this introductory setting we will restrict ourselves to networks that utilize Rectified

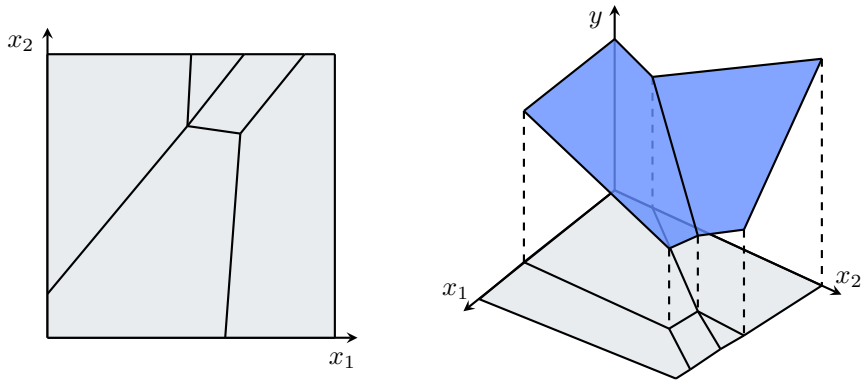

Figure 1.1: Decomposition of a piecewise linear neural network $f : \mathbb{R}^2 \to \mathbb{R}$ into regions of affine transformations. (Left) The regions in the x_1x_2-plane with the region borders represented by solid lines. For simplicity, the regions are only illustrated in an axis-parallel rectangle. (Right) A cross-section of the local affine transformations associated with the regions.

Linear Units (ReLU) as activation functions. In this case

$$g(x) = \max(0, x).$$

For these networks with ReLU activation functions, the regions which constitute the decomposition of the input space into sets where the network is completely described by an affine relation are known as *activation regions*. These are obtained by sending an input through the network and observing the resulting binary pattern (known as *activation* patterns) indicating which neurons that are active (output > 0) and inactive (output ≤ 0) in the hidden layers, hence the name activation region. Mathematically, for a given x we can define an activation region $\mathcal{R}_x$ as

$$\mathcal{R}_x := \{\tilde{x} \in \mathbb{R}^{n_0} : 1\{f^l(\tilde{x}) > 0\} = 1\{f^l(x) > 0\}, l = 1, 2, ..., n_{n-1}\}$$

where the inequalities and indicator function 1 is assumed to act componentwise. For the rest of this thesis, if a property is valid for all activation regions, we drop the subscript x and simply refer to a general region as $\mathcal{R}$. That the activation regions form the above mentioned decomposition of the input space can be understood in the manner that switching a neuron from off to on (on to off) entails engaging (disengaging) another non-linearity, thus the network will with probability 1 be non-linear for any section in the input space that encompasses different activation regions [Hanin and Rolnick (2019b)].

While not strictly piecewise linear, we will additionally recognize networks where a non piecewise linearity is only used in the output layer, e.g., sigmoid, softmax or similarly, as piecewise linear. These final activation functions often only serve to map the output into a predefined range and can in many cases be removed with minimal ramifications. Such a network can thus easily be converted to an equivalent piecewise linear network by the removal of the activation function on the output layer and all analysis can be performed on that equivalent piecewise linear network instead.

In this thesis we mainly focus on the properties of the affine transformation associated with each region, and only showcase minor results for the regions themselves. This restriction is mainly done due to the fact that previous research into the regions has already extensively detailed many of their properties and behaviour. For example, the regions have been shown to be convex polytopes [Hanin and Rolnick (2019b)], to act as hash encoders for the training set [He et al. (2021)], used to reverse-engineering neural networks [Rolnick and Kording (2020)], utilized for formal verification and robustness against adversarial inputs [Botoeva et al. (2020); Croce et al. (2019)] and have been connected to the interpretability of the network [Xu et al. (2021)].

On the other hand, little is known about the behaviour of the affine transformation. We thus focus our investigation on this aspect of the piecewise linear networks and an excerpt of some natural inquires for these transformations can be seen below:

- How different can the affine transformation be between two neighbouring regions?
- What is a typical rank of the affine transformation?
- How are shapes preserved when mapped under them?
- How are the properties of the affine transformations related to the training algorithm?

In this thesis we seek to explore and answer some of these inquires. Answering these will help characterize piecewise linear networks and subsequently facilitate any analysis into the behaviour of deep learning in this piecewise linear regime.

2 Geometric exploration

Since the task that we have undertaken is exploratory in nature, some experiments that we have performed have given results that are interesting but not significant enough to stand on their own for a publication. While not major, these results are a natural starting point for setting the stage for the larger investigations that have been performed and showcasing how the geometry behaves in different situations. We will thus present some of them in the ensuing sections.

2.1 Geometric properties of the activation regions

In order to use the activation regions to characterize the networks we will first need some suitable measure to characterize the activation regions themselves and their properties. For example, attributes such as small, big or elongated are among some geometric properties of the regions that could serve to delineate the differences between the piecewise linear networks, possibly providing a characterization between the networks that can successfully learn a task and those that cannot. We thus start with detailing two possible geometric measures for this task, the volume of the region and the distance from a point to the nearest region edge.

2.1.1 Volume of regions

A natural measure of the regions to consider is that of the *volume* of the regions. Unfortunately this measure is deficient for the task of understanding the geometric properties for a wide variety of network structures. This deficiency can easily be understood through two properties:

- Obtaining the volume of a single convex polytope in $\mathbb{R}^n$ scales as $O(n^5)$, making it unfeasible to obtain the volume of a region for the large input dimensions present in many deep learning problems [Kannan et al. (1997)].

- For networks where the number of neurons is smaller than the input dimension, the volume of every activation region is infinite. Thus the measure will provide no information for these networks. This follows from the fact that each wall of an activation region is given by the set where a neuron is switched from on to off or vice versa. Since $n_0 + 1$ hyperplanes are needed to constrict a region of finite volume in $\mathbb{R}^{n_0}$, each region will have an infinite volume if there are fewer neurons than the dimension of the input space, meaning $n_0 \geq \sum_{i=1}^{n-1} n_i$.

The volume is thus not a suitable measure to facilitate a characterization of the piecewise linear networks.

2.1.2 Distance to the edge of a region

A measure that we opt for and that is more feasible is that of the closest distance d^* from a point x to the edge of the associated region $\mathcal{R}_x$. This measure is also suitable since it can allow us to analyze geometric differences that can arise between two different sets of points, e.g., between the training and validation set.

The first question is how to calculate this measure in an efficient manner. Since the activation region is defined by a specific activation pattern, one way to obtain the analytical distance to the region edge is to simply calculate the distance d from x to all neuron decision boundaries. The smallest obtained distance will be d^*, i.e.,

$$d^* = \min_{\substack{l=1,2,\ldots,n-1 \\ i=1,2,\ldots,n_l}} \text{dist}\Big(x, \{x \in \mathbb{R}^{n_0} : \big(W_l f^{l-1}(x) + b_l\big)_i = 0\}\Big), \tag{2.1}$$

which can be seen in Figure 2.1 (left). While this distance to the region edge can be obtained in a straightforward manner, the time complexity scales as $O(\sum_{l=1}^{n-1} n_l)$, making it expensive to calculate for larger networks. For simplicity, in this introductory part of the thesis we will restrict ourselves to smaller networks and thus (2.1) will suffice to demonstrate the properties of interest. However, if one wishes to analyze the distance to the nearest region edge for larger networks then some form of stochastic approximation of (2.1) is

necessary to keep the computational burden to a minimum.[1]

An immediate possible application emerging from this measure is that of detecting data points where the function has to adapt quickly to provide a good fit, e.g., for mislabeled data points enclosed by data points of the correct class. For such data points, the required quick adaptation of the function to the mislabeled point compared to that of the surroundings is likely to cause the function to need multiple different affine transformations to ensure a good fit, necessitating crossing multiple activation regions. One can thus hypothesize that such mislabeled data points are situated closer to the region boundary. A hypothesis that we explore in the subsequent section.

2.1.3 Detecting mislabeled data points

Building on the hypothesis that mislabeled points should be closer to the region boundary than that of 'normal' points, we set up an experiment where we first mislabel 100 data points in the training set of MNIST and then train a network on the full MNIST training set consisting of 60000 points. Subsequently we compare the distance to the nearest region edge for these 100 points and 500 randomly chosen 'normal' points[2] and compare their resulting distributions, see Figure 2.1 (right).

As can be seen from the histogram, the mislabeled data points are situated in the lower end of the full distribution, giving some evidence to the validity of the hypothesis. Unfortunately it can also be seen that there is a substantial amount of correctly labeled points which are located close to the region boundary, making it impractical to use this scheme to detect possible mislabeled data points.

2.2 Geometric properties of the affine transformations

The first question that we have to ask ourselves is how we explore the properties of the affine mapping in every activation region. Since the network f is locally

[1] It is not difficult to envision such a stochastic approximation scheme. Simply sample points on concentric spheres of increasing radii around the point of interest and when a sampled point obtain a different activation pattern one has an upper bound on d^*.

[2] Normal might be an ambiguous term since there most likely already exists mislabeled data points in the original MNIST data set.

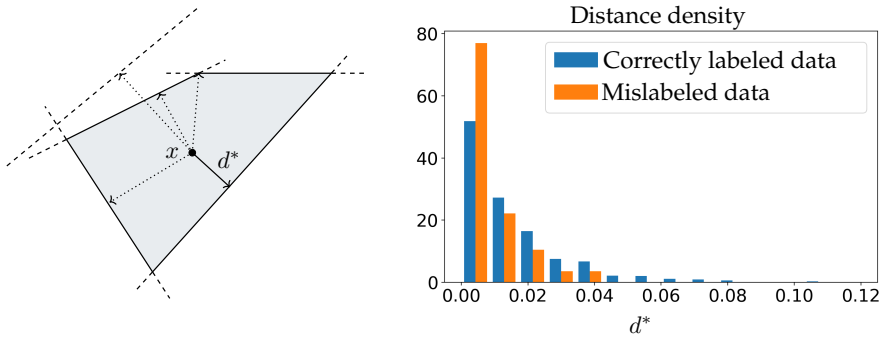

Figure 2.1: Distance to the closest region edge. (Left) Illustration of the nearest region distance d^*. The dotted lines are the distances that has to be evaluated in equation (2.1). (Right) Histogram over the differences in distances to the nearest region edge for mislabeled and correctly labeled data points. The mislabeled histogram is created from 100 data points and the correctly labeled histogram from 500 data points. Experimental details associated with the plots can be found in Appendix A

affine in every activation region R, meaning that there exists $W_R \in \mathbb{R}^{n_c \times n_0}$ and $b \in \mathbb{R}^{n_c}$ such that $f(x) = W_R x + b, \forall x \in R$, we can explore the properties of the mapping by exploring the properties of W_R. The main properties that we are interested in is the rank and singular values of W_R.

Throughout this section we will use the fact that the affine transformation W_{R_x} for a region R_x can be obtained as

$$W_{R_x} = W_n Z_{n-1}(x) W_{n-1} \cdots Z_2(x) W_2 Z_1(x) W_1 \tag{2.2}$$

where $Z_i(x)$ is a diagonal boolean matrix indicating which neurons in layer i that are active when passing input x through the network. To simplify the notation, if a property is valid for all $R \in \mathcal{R}$, we will simply write $Z_i(x)$ as Z_i. We can now proceed with our investigation into the rank and singular values of W_R.

2.2.1 Rank of the affine transformation

The rank of the affine transformation gives information regarding the dimension of the space that the input data points are mapped to. A low rank could indicate that the training procedure is not optimal, and that the network is not fully utilizing the available capacity to solve the task at hand. For compression tasks the rank can also indicate the amount of superfluous information that is

removed from the input. It is therefore of interest to investigate the rank and its behaviour further.

Theoretical properties. The first step in analyzing the rank is to have an efficient way of obtaining it. At initialization of the network, the rank of the affine transformation W_R associated with a region R can easily be obtained as

$$\text{rank}(W_R) = \min(n_0, n_c, \min_i \text{trace}(Z_i)), \tag{2.3}$$

meaning that we only need to check which quantity that is the smallest between the input dimension, the output dimension and the number of active neurons at every layer in order to obtain the rank. This follows since the effect of Z_i in (2.2) is to simply zero out rows of the corresponding matrices W_i, leaving us with a product of random matrices which can be deduced to have a rank given as (2.2), see Appendix A for a proof. While this rank property can be proven to hold at initialization, we make the assumption that the learning trajectory is such that the equality remains valid even when updating the parameters during SGD, allowing us to analyze the rank and its behaviour during training.

Empirical observations. As mentioned previously, the rank of the affine transformation might indicate when the training is not proceeding as intended, providing us with a possible tool for debugging the training procedure and facilitating the model construction process. One such example where the behaviour of the rank can assist an analyst in the model construction process can be seen below in Figure 2.2 where we train a network with two hidden layers with 200 neurons each on FashionMNIST [Xiao et al. (2017)]. We train the network with a learning rate λ that is too large for the training dynamics to properly converge and compare it to a learning rate for which the training proceeds smoothly. From the loss and accuracy curves we can see that after a certain amount of epochs the network trained with a high learning rate collapses and that the model predictions are of little use beyond this point. While this information is available from the loss curves, it is not visible as early as it is for the rank trajectory. Comparing the low learning rate setting where the rank stays maximal throughout the entire training to the high learning rate setting where the rank starts decreasing immediately we can quickly understand that something is amiss in the high learning rate setup. Since the environmental impact and cost of training deep learning models has substantially increased in the last decade [Strubell et al. (2019)], any scheme that can allow an analyst to cancel a flawed training run in advance can be of benefit.

However, to make this a practical scheme one would need to obtain solid evidence that this behaviour does not occur for non-flawed runs, thus ensuring that no well-performing models are discarded.

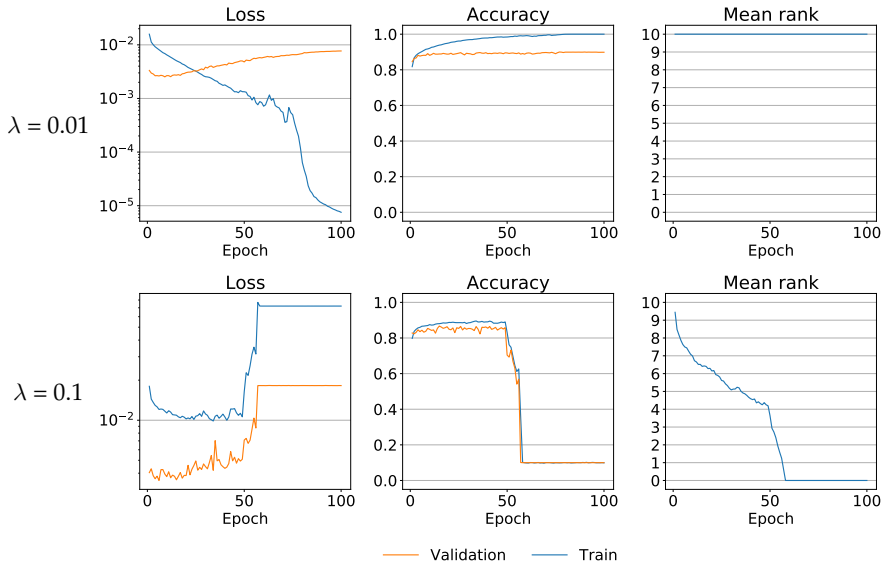

Figure 2.2: Difference in rank between two different learning rates. (Top row) The training evolution for a network with learning rate λ equal to 0.01. The rank is stable throughout the training. (Bottom row) The training evolution for a network with λ equal to 0.1. The rank is decreasing throughout most of the training, making it clear that it might be wise to cancel the training procedure at an early stage. Experimental details associated with the plots can be found in Appendix A.

2.2.2 Singular values of the affine transformations

The singular values of W details the extent of how far away points can be mapped under W. The values thus give insights into if all points in a region are mapped to a similar location or if they can be spread out over space, allowing us to obtain a more detailed understanding of the geometry of the network mapping. We explore the singular values of W both through theoretical analysis and empirical experiments.

Theoretical properties. We start this exploration by first studying how the singular values can vary as we move between two adjacent activation regions R and $\tilde{R}$, where with adjacent we mean that the associated activation patterns for R and $\tilde{R}$ only differ with one element.

Returning to equation (2.2) and assuming that the activation patterns of R and

$\tilde{R}$ differ only because of the k:th neuron in the l:th layer, we have that we can write W_R and $W_{\tilde{R}}$ as

$$W_R = W_n Z_{n-1} W_{n-1} \cdots Z_l \cdots Z_2 W_2 Z_1 W_1 \tag{2.4}$$

$$W_{\tilde{R}} = W_n Z_{n-1} W_{n-1} \cdots \tilde{Z}_l \cdots Z_2 W_2 Z_1 W_1 \tag{2.5}$$

where Z_l and $\tilde{Z}_l$ only differ in the k:th element on the diagonal. Using this fact we can write either of the matrices as a *perturbed* version of the other, meaning that $W_R = W_{\tilde{R}} + E$ where

$$E = W_n Z_{n-1} W_{n-1} \cdots (Z_l - \tilde{Z}_l) \cdots Z_2 W_2 Z_1 W_1. \tag{2.6}$$

This identity allows us to make use of perturbation theory for singular values [Stewart (1990)] to obtain a bound on the differences of the singular values as

$$\sqrt{\sum_i (\sigma_i - \tilde{\sigma}_i)^2} \leq ||E||_F, \tag{2.7}$$

where σ_i denotes the i:th singular value of W_R and similarly for $\tilde{\sigma}_i$ and $W_{\tilde{R}}$. While expression (2.7) does not immediately yield itself to a theoretical analysis for general networks, by restricting ourselves to the single layer fully connected case we can proceed with more ease. For these networks the squared Frobenius norm of the perturbation is given as

$$||E||_F^2 = \sum_i \sum_j \left(W_2\right)_{k,j}^2 \left(W_1\right)_{k,i}^2, \tag{2.8}$$

where W_2, W_1 are the weights of the inward and outward connections to the hidden layer, and thus for these networks we can perform an analysis with more ease. A straightforward application of this identity is to analyze how the singular values vary between adjacent regions at initialization of the network. Many of the commonly used initialization schemes initialize the weights of the network independently according to a distribution with zero mean and variance proportional to the number of neurons in the previous layer, i.e., $\sigma^2 = C/n_{l-1}$ for weights in layer l for some $C > 0$ [Glorot and Bengio (2010); He et al. (2015)]. Thus for these networks, the expected difference of the

singular values can be upper bounded as

$$\mathbb{E}[\sum_i(\sigma_i-\tilde{\sigma}_i)^2] \leq \mathbb{E}\left[||E||_F^2\right] = \sum_i\sum_j \mathbb{E}\left[\left(W_2\right)_{k,j}^2\left(W_1\right)_{k,i}^2\right] \tag{2.9}$$

$$= \sum_i\sum_j \mathbb{E}\left[\left(W_2\right)_{k,j}^2\right]\mathbb{E}\left[\left(W_1\right)_{k,i}^2\right] \tag{2.10}$$

$$= \sum_i\sum_j \frac{C}{n_1}\frac{C}{n_0} \tag{2.11}$$

$$= n_0 n_c \frac{C}{n_1}\frac{C}{n_0} \tag{2.12}$$

$$= \frac{C^2 n_c}{n_1}. \tag{2.13}$$

From this expression we can see that the expected difference decreases when more neurons are added to the hidden layer, giving an indication that adjacent regions might behave similarly at initialization for large networks[3].

While the previous analysis allows us to get a better grasp on the local variations of piecewise linear networks, we still lack a global perspective of the behaviour of the network. Further restricting ourselves to the setting where $n_0 = n_c = 1$, we can partly resolve this issue by utilizing information regarding the number of activation regions that one can encounter along any interval. Knowing how the singular values of the regions can vary as we go between them and additionally how many regions we cross in a given interval will allow us to obtain a better understanding of how the function varies in that interval.

If we introduce the random variable

$$N : \text{Number of activation regions in a unit interval} \tag{2.14}$$

then recent work in the setting $n_0 = n_c = 1$ proves that at initialization we will have $\mathbb{E}[N] \approx \#\{neurons\}$ [Hanin and Rolnick (2019a)]. Utilizing this fact and making the additional assumption that the distribution for N is sharply peaked at the mean, i.e., $P(N = \mathbb{E}[N]) \approx 1$, then we can bound the squared

[3]Unfortunately, except for the case where $n_0 = n_c = 1$, replacing 'might behave' with 'will behave' requires additionally that the singular vectors should be similar for large networks. This is more difficult to analyze theoretically.

slope difference for two regions $R, \tilde{R}$ a unit distance away from each other as

$$\mathbb{E}[(\sigma - \tilde{\sigma})^2] \leq \mathbb{E}\Big[\sum_{j=1}^{\text{N - 1}} 2[(\sigma_j - \tilde{\sigma}_{j+1})^2\Big] \tag{2.15}$$

$$\approx \sum_{j=1}^{n_1-1} 2\mathbb{E}[(\sigma_j - \tilde{\sigma}_{j+1})^2] \tag{2.16}$$

$$\leq 2(n_1 - 1)\frac{C^2}{n_1} \tag{2.17}$$

$$= 2C^2 - 2\frac{C^2}{n_1}. \tag{2.18}$$

Thus we can see that there is a constriction on how different the average slope at initialization in two regions a unit distance away from each other can be, giving a further geometric understanding of neural networks in the 1d-regression setting.

Empirical observations. Given that (2.13) and (2.18) only provides upper bounds on the difference of the singular values, it is of interest to compare the bounds with empirically obtained estimates to see how tight they they are. Thus we perform an empirical experiment where we initialize a single layer neural network in PyTorch [Paszke et al. (2019)] (which uses Xavier initialization where the components of W_l are sampled independently according to $U[-1/\sqrt{n_{l-1}}, 1/\sqrt{n_{l-1}}]$, giving $C = 1/3$ [Glorot and Bengio (2010)]) with $n_c = 10$, vary the number of neurons in the hidden layer and obtain the squared difference of the singular values. The comparison can be seen in Figure 2.3 where in both plots we can see that the bounds are relatively loose. The bound (2.13) gives an indication for how the difference evolves as the network grows while (2.18) can be seen to only provide a rough upper bound and no additional information.

2.3 Conclusion

From these minor results, regarding the distance to the closest region and the behaviour of the rank and singular values of the affine transformations, we can see how these geometric properties can allow us not only to understand the piecewise functions better, but also to envision schemes that can facilitate the usage of deep learning.

We now move on to the more significant results and applications which we

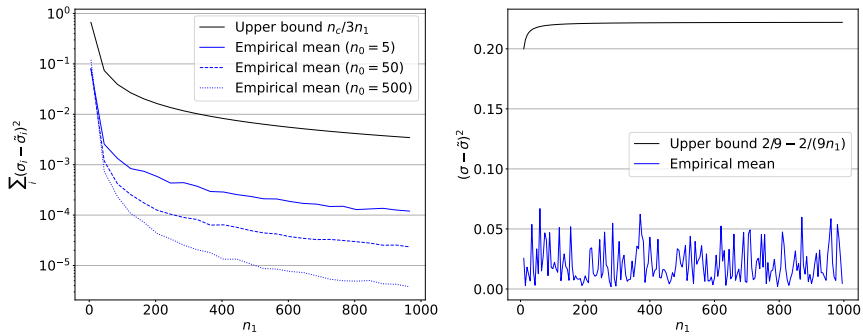

Figure 2.3: How the difference between the singular values evolve as the network grows wider. (Left) The singular value difference between adjacent regions for a single layer network with $n_c = 10$. (Right) The singular value difference between regions a distance 1 from each other for a single layer network with $n_0 = n_c = 1$. It can be seen that both bounds are relatively loose. Experimental details associated with the plots can be found in Appendix A.

have obtained, compiled into the four papers on which this thesis is based.

3 Overview of papers

In this section we provide an introduction and conclusion to each paper. While **Paper II** and **Paper IV** are related to general properties and schemes for neural networks, **Paper I** and **Paper III** are more focused on applying neural networks to solve problems in the automotive industry.

3.1 Paper I

3.1.1 Introduction

One of the main difficulties that can occur when trying to utilize deep learning for a given task is that of ensuring that the method works as intended over a wide range of scenarios. Measuring the validation loss can give an indication of the efficacy of the model but such a measure can fail to give an indication of how it will work in deployment when new unknown scenarios, unlike those appearing in the train or validation set, will be presented to the model.

Since the objective of stochastic gradient descent is to reduce the loss over a given batch containing different scenarios, it is reasonable to assume that the scenarios that are commonly occurring in our training set will have a greater effect on the trajectory of the descent procedure. We thus work under the assumption that the descent will yield a model that will be able to perform well for the scenarios that are abundant in the training data and potentially struggle when encountering sparsely occurring scenarios. For example, consider a self-driving bus as in Figure 3.1 where a component in the vehicle is meant to display bounding boxes around objects of interest, e.g., traffic signs, vehicles and pedestrians. If such a component relies on deep learning then it it is likely that the component will fail to output bounding boxes around objects that occur sparsely in the training data, potentially leading to catastrophic consequences.

Here this is visualized as the failure to detect the cyclist in the bottom right, the cause potentially being the lack of cyclists or pedestrian crossings in the training data.

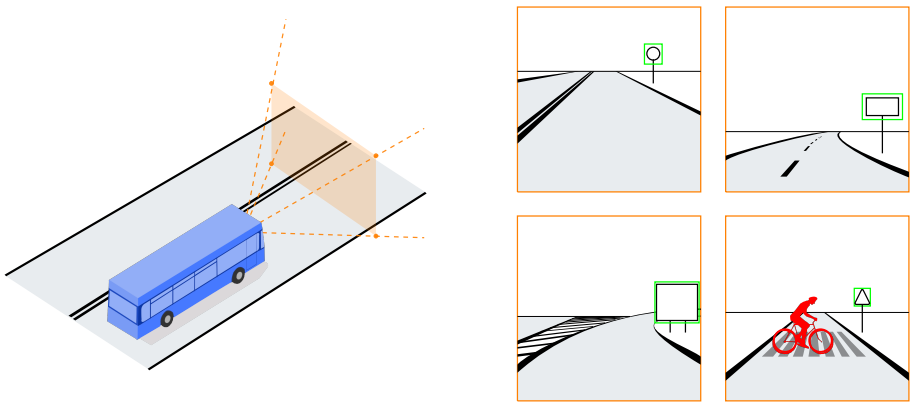

Figure 3.1: Illustration of a potential failure mode when components rely on deep learning, here showcased as the failure of a bounding box detector. (Left) A self-driving bus with its corresponding field of vision. (Right) The different scenarios the bus encounters and the bounding boxes represented in green. It can be seen that the system manages to enclose traffic signs but fails to create a bounding box for the cyclist in the bottom right. Such a failure could be attributed to a lack of cyclists or pedestrian crossings in the training data.

With the data sets required to obtain accurate models growing ever larger, it is becoming infeasible to manually label which scenarios that are abundant and which that are sparsely occurring in the training data, making that a difficult path to conclude that the sparsity of a scenario likely contributed to the failure. While obtaining labels from samples is infeasible, simulation based methods can allow us to approximately obtain samples from labels, opening up the way to generate novel scenarios and discover failures modes for real driving scenarios by exploring the models behaviour on simulated samples.

In **Paper I** we thus postulate that these novel scenarios are likely to be regarded as outliers by the model, converting the problem of detecting possible sparsity issues of the training data to one of outlier detection, see Figure 3.2.

Since these novel scenarios can be generated with ease through simulation, we circumvent the expensive annotation of the training data and can identify scarce scenarios, ultimately effectivizing further data collection schemes.

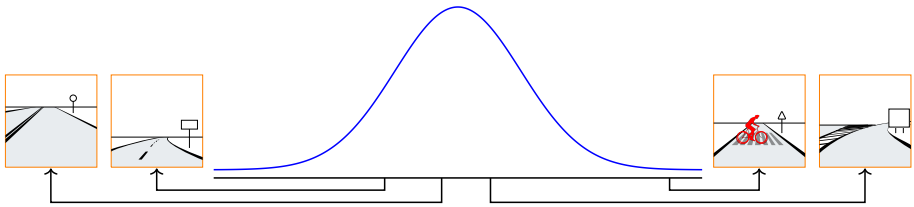

Figure 3.2: Illustration of the novelty hypothesis. Data points representing novel scenarios can be represented as outliers, converting the problem of identifying possible failure modes to one of outlier detection. Here the blue curve represents the empirical data distribution obtained from the training set. The cyclist scenario can clearly be seen to be an outlier compared to the other scenarios.

3.1.2 Conclusion

By working with piecewise linear networks related to the VGG-networks [Simonyan and Zisserman (2015)] and with a data set of circles and squares, we show that by utilizing outlier detection techniques for neural networks we can get an indication of sparsity issues in the training data, confirming the posed novelty hypothesis. Further, by specifying a distribution of expected scenarios in terms of low level interpretable features and comparing the support of the empirical training data distribution to that of the distribution of expected scenarios, we are additionally able to obtain a global interpretable understanding of the characteristics of the training data. This information can facilitate further data collection schemes by steering collection efforts towards sparsely occurring scenarios.

3.2 Paper II

3.2.1 Introduction

One of the main considerations that has to be taken into account when performing any kind of machine learning is that of overfitting. While measuring the validation loss can give an indication of overfitting, such a measure does not necessarily prevent it from occurring. One common way of counteracting overfitting and simultaneously promoting generalization is that of enforcing a restriction on a suitable geometric property of the functions that one considers, where suitable should be interpreted as that there exists a connection between the property and the inclination of the function to overfit.

While the existence of such a property might appear obscure at first, in 1d-regression it is not difficult to realise that the derivative is a geometric property that is associated with the generalization, and that enforcing a penalty on the magnitude of the derivative is likely to improve generalization for a wide range of tasks. This fact is visualized in Figure 3.3 where we see that the generalization for polynomial regression is improved when a penalty on the magnitude of the weights is added to the loss function, consequently restricting the size of the derivative of the estimators.

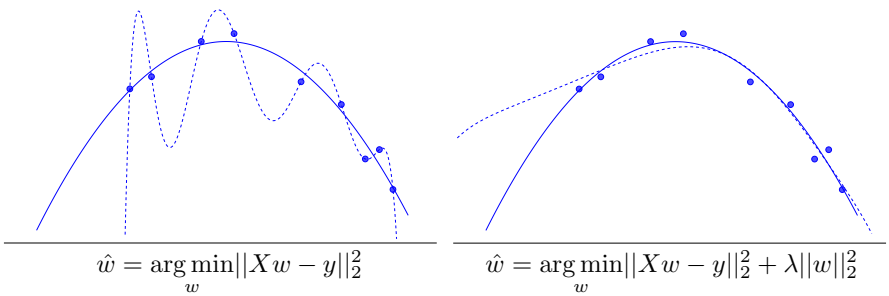

Figure 3.3: Illustration showcasing that the derivative is a geometric property related to generalization for 1d-regression. The data generating function is given by the second order polynomial drawn in solid blue and the dashed line indicates the fit an eighth order polynomial with coefficients $\hat{w}$. (Left) Fitting the polynomial without any restrictions (Right) Fitting the polynomial with an added penalty on the magnitude of the weights. The generalization improves as the penalty is added indicating that the derivative is related to the generalization.

While the derivative is clearly connected to the generalization for 1d polynomial regression, it is not immediately clear which geometric properties that are tied to generalization for neural networks employed in high dimensional tasks such as image classification or bounding box detection.

Understanding which geometric properties that are associated with the generalization can allow for the development of new regularization schemes and will additionally facilitate the understanding of the high-dimensional behaviour of neural networks. While such a property might be elusive for general networks, by restricting ourselves to piecewise linear networks we can employ the geometric perspective and understand the functions in terms of the constituent activation regions and the corresponding affine transformations. In this setting, a natural choice for geometric properties to consider are those that describe the behaviour of the local mappings W_R. Further basing our intuition on the one-dimensional regression case, we can realise that the mappings ability to quickly adapt to variations is a property that can induce overfitting. While for

a high-dimensional mapping this property is not completely captured by any single number, the magnitude of the increase in the direction of fastest increase captures an aspect of the mappings ability to adapt to sudden variations in the data, and it is thus of interest to explore its relation with the generalization of the network.

We refer to this property as *slope* [Johansson et al. (2021)] and subsequently define it as:

Definition 1. *Given a continuous function* $f : \mathbb{R}^{n_s} \to \mathbb{R}^{n_c}$ *and some* p *with* $1 \leq p \leq \infty$*, we define its slope (or p-slope) at* $x \in \mathbb{R}^{n_s}$ *as*

$$\text{Slope}_f(x) = \sup_{v \in B^*} \left(\lim_{t \downarrow 0} \frac{||f(x+tv) - f(x)||_p}{t} \right)$$

where $||\cdot||_p$ *denotes the p-norm*[1]*, the limit is taken over positive* t*, and*

$$B^* = \{v \in \mathbb{R}^{n_s} : ||v||_p = 1\}.$$

The slope is undefined unless the limit exists for all $v \in B^*$*.*

In **Paper II** we investigate how the slope varies as we train different networks to generalize on different data sets and additionally explore its properties as the data set is modified.

3.2.2 Conclusion

By training multiple network architectures on different classification tasks, we could observe that the slope is a geometric property that have an association with the generalization. For all networks the slope grows continuously during training which signals that when the network inevitably overfits the slope will be larger than that compared to a generalizing network obtained at an earlier epoch through early stopping. Additionally, for all generalizing networks the slope is relatively stable to changes in the network width and only possess a weak dependence on the depth of the network, indicating that all networks express the underlying function in a similar manner.

We thus see evidence that controlling the slope can be another way of promoting generalization for neural networks.

[1] $||x||_p = \left(\sum_i |x_i|^p\right)^{1/p}$

3.3 Paper III

3.3.1 Introduction

A fundamental task in ensuring the safety of modern vehicles lies in ensuring that the integrated software behaves as intended. For example, if the automatic brake system engages erroneously it can have the effect of disrupting traffic and potentially leading to a fatal collision, making it a crucial task to verify its soundness before the vehicle is put on the market. This ensurance is often an exercise in exposing the vehicle to a wide array of scenarios and verifying that the software acts as intended under all circumstances. With the software interpreting the surrounding world through real-valued time-series containing information regarding the state of the vehicle, e.g., the evolution of the vehicle speed and engine speed, exposing the software to a wide variation of scenarios corresponds to exposing the software to a wide range of varying time-series which captures a possible driving trajectory for the vehicle.

While arguably the most straightforward way of obtaining these signals is to drive the vehicles around and record the time-series resulting from the interplay between all the vehicle properties, it is not a practical approach and is resource intensive. Consider the scenario in Figure 3.4 where we want to record the properties of a bus when driving along a winding route. Assuming that our software under test only relies on the evolution of the vehicle speed, engine speed and selected gear we selectively focus our attention on these three properties. While such a test drive along the route will initially allow us to verify the function of our software under this scenario, a question naturally arises what we do if we want to extend upon the scenario and test our software under slightly different conditions, e.g., recording the state properties of the bus as it is carrying a different amount of passengers or as the friction with the road is changed. Sending the vehicle out in the field again along the same route over and over to record slight variations of the same scenario is clearly a scheme that will be ineffective, cost-consuming and polluting.

In **Paper III** we thus seek to improve on this methodology. We utilize data from previous test drives for buses to create a Generative Adversarial Network which is able to transform a simple specification of a test to a fully fledged time series adhering to that given specification, see Figure 3.5.

Such a scheme circumvents the expensive testing routine described above and instead allows engineers to obtain realistic tests without access to a physical vehicle, effectivizing the testing procedures and reducing the emissions of greenhouse gases.

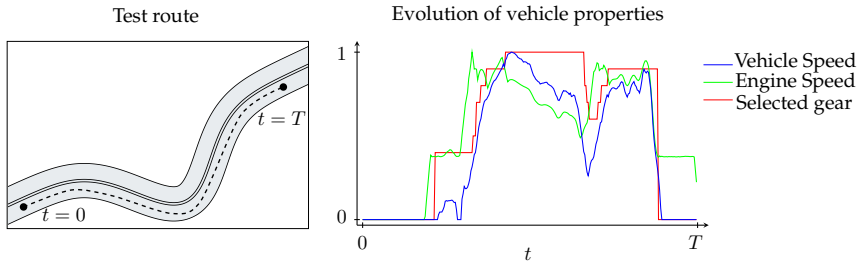

Figure 3.4: Test route and the associated evolution of the vehicle properties over time for a bus. (Left) The test route (dashed line) that the bus drives along. (Right) The resulting evolution of the vehicle properties captured in terms of the vehicle speed, engine speed and selected gear. The relevant information to test our software is captured in the three time series. All signal values have been normalized to lie between 0 and 1.

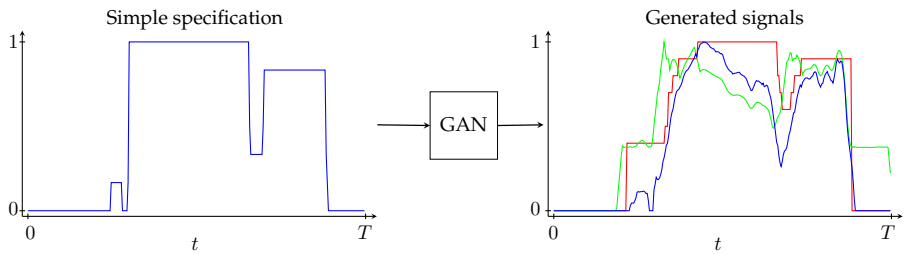

Figure 3.5: Conversion of a specification to a realistic test scenario. (Left) The specification detailing our desired test. Only the desired variations in the vehicle speed is specified. (Right) The transformed specification, consisting of the three signals of interest and adhering to the given specification. The conversion is done by training and utilizing a GAN.

3.4 Conclusion

By training a GAN[2] on previously collected test scenarios we demonstrate that one can obtain realistic fully-fledged signals from simple specifications in the form of one-dimensional time series. By constructing a expansion module for the network we additionally showcase that partial specifications are possible, reducing the requirement on the engineer to specify the duration of the entire test scenario. This entire scheme facilitates the testing procedure and enables

[2]While this network is not completely piecewise linear due to the usage of instance normalization layers [Ulyanov et al. (2016)], it can still be argued to be well described by the piecewise linear theory since the predominant non-linear transformations are piecewise linear and thus control the training dynamics.

a shift from hardware based testing setup to one more reliant on software, consequently opening a path for more effective and environmentally greener automotive tests.

3.5 Paper IV

3.5.1 Introduction

While ensuring that our neural networks generalize is one of the key aspects model deployment, ensuring that the models are robust to noise can be an equally important aspect. Verifying that our models can provide accurate predictions in the presence of both natural and adversarial noise can in some settings be a precondition to model deployment, for example in the self-driving vehicle sector where one can immediately realise the dangers with a self-driving car relying on a model susceptible to adversarial attacks.

From the results in **Paper II** indicating the slope as a factor for generalization and with previous research indicating that controlling the slope promotes robustness of the network [Yoshida and Miyato (2017)], we sought to develop a scheme that can regularize the slope in an efficient manner. While previous techniques were able to control the slope to some extent, they do so in the form of upper bounds, lacking a fine-grained control over the quantity of interest. We thus wanted a more precise scheme that targets the slope directly while at the same time not being too computationally expensive to use.

Using the same identity in equation (2.2) we show that one can obtain an efficient and precise regularization scheme that targets the slope for piecewise linear networks where one sends an input forward and *backward* through the network which allows for a subsequent power iteration technique to estimate the slope, see Figure 3.6 for a visualization of the procedure and how it relates to a regular forward-pass.

3.5.2 Conclusion

By utilizing our proposed scheme to regularize the slope we demonstrate that we obtain an improved test accuracy over different data sets compared to the methods that only rely on upper bounding the desired quantity.

When it comes to robustness we could observe that working with our scheme

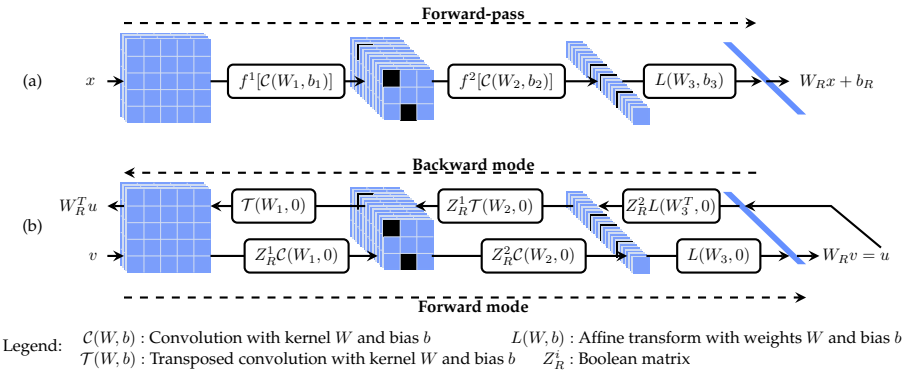

Figure 3.6: The difference between a regular forward-pass and the forward and backward modes for a two hidden layer network. (a) A regular forward-pass of x through the network. Each box showcases the operation that maps the input between the layers. The black squares indicate the neurons mapped to zero by the ReLU activation functions f^i. (b) The forward and backward modes used to estimate the slope $||W_R||_2$. An input v is sent through the network to yield u whereupon u is sent backwards through the network. Adapted from Figure 1 in **Paper IV**.

maintains a strong safeguard against adversarial noise, with the method for certain data sets and attacks achieving significantly superior robustness compared to other methods.

Bibliography

Botoeva, E., Kouvaros, P., Kronqvist, J., Lomuscio, A., and Misener, R. (2020). Efficient verification of relu-based neural networks via dependency analysis. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 3291–3299. AAAI Press.

Croce, F., Andriushchenko, M., and Hein, M. (2019). Provable robustness of relu networks via maximization of linear regions. In Chaudhuri, K. and Sugiyama, M., editors, *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pages 2057–2066. PMLR.

Feng, X. and Zhang, Z. (2007). The rank of a random matrix. *Applied Mathematics and Computation*, 185(1):689–694.

Gholamalinezhad, H. and Khosravi, H. (2020). Pooling methods in deep neural networks, a review. *CoRR*, abs/2009.07485.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, D. M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. C., and Bengio, Y. (2014). Generative adversarial networks. *CoRR*, abs/1406.2661.

Hanin, B. and Rolnick, D. (2019a). Complexity of linear regions in deep networks. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019,*

Long Beach, California, USA, volume 97 of *Proceedings of Machine Learning Research*, pages 2596–2604. PMLR.

Hanin, B. and Rolnick, D. (2019b). Deep relu networks have surprisingly few activation patterns. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 359–368.

He, F., Lei, S., Ji, J., and Tao, D. (2021). Neural networks behave as hash encoders: An empirical study. *CoRR*, abs/2101.05490.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1026–1034. IEEE Computer Society.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Bach, F. R. and Blei, D. M., editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org.

Isola, P., Zhu, J., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5967–5976. IEEE Computer Society.

Johansson, A., Engsner, N., Strannegård, C., and Mostad, P. (2021). Slope and generalization properties of neural networks. *CoRR*, abs/2107.01473.

Kannan, R., Lovász, L., and Simonovits, M. (1997). Random walks and an $o^*(n^5)$ volume algorithm for convex bodies. *Random Struct. Algorithms*, 11(1):1–50.

Kong, J., Kim, J., and Bae, J. (2020). Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In Fürnkranz, J. and Joachims, T., editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 807–814. Omnipress.

Otter, D. W., Medina, J. R., and Kalita, J. K. (2021). A survey of the usages of deep learning for natural language processing. *IEEE Trans. Neural Networks Learn. Syst.*, 32(2):604–624.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E. Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Redmon, J., Divvala, S. K., Girshick, R. B., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 779–788. IEEE Computer Society.

Reed, S. E., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., and Lee, H. (2016). Generative adversarial text to image synthesis. In Balcan, M. and Weinberger, K. Q., editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1060–1069. JMLR.org.

Roger, V., Farinas, J., and Pinquier, J. (2020). Deep neural networks for automatic speech processing: A survey from large corpora to limited data. *CoRR*, abs/2003.04241.

Rolnick, D. and Kording, K. P. (2020). Reverse-engineering deep relu networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 8178–8187. PMLR.

Schmidhuber, J. (2020). Generative adversarial networks are special cases of artificial curiosity (1990) and also closely related to predictability minimization (1991). *Neural Networks*, 127:58–66.

Shahriar, S. (2021). GAN computers generate arts? A survey on visual arts, music, and literary text generation using generative adversarial network. *CoRR*, abs/2108.03857.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Stewart, G. W. (1990). Perturbation theory for the singular value decomposition. In *IN SVD AND SIGNAL PROCESSING, II: ALGORITHMS, ANALYSIS AND APPLICATIONS*, pages 99–109. Elsevier.

Strubell, E., Ganesh, A., and McCallum, A. (2019). Energy and policy considerations for deep learning in NLP. In Korhonen, A., Traum, D. R., and Màrquez, L., editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3645–3650. Association for Computational Linguistics.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1–9. IEEE Computer Society.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. S. (2016). Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747.

Xu, S., Vaughan, J., Chen, J., Zhang, A., and Sudjianto, A. (2021). Traversing the local polytopes of relu neural networks: A unified approach for network verification. *CoRR*, abs/2111.08922.

Yoshida, Y. and Miyato, T. (2017). Spectral norm regularization for improving the generalizability of deep learning. *CoRR*, abs/1705.10941.

A Appendix

A.1 Proofs

Proof (Proof of equation (2.3)).

We will not prove equation (2.3) in all generality but restrict ourselves to real independent random element matrices which we define below.

Definition 2. *A real independent random element (RIRE) matrix is a real matrix A where all elements A_{ij} are distributed independently and identically according to an absolutely continuous probability measure w.r.t the Lebesgue measure.*

While this is a restriction, it encompasses all default initialization schemes for neural networks, e.g., used in PyTorch.

Additionally, for the proof we will need two Lemmas.

Lemma 1. *If A is a RIRE matrix then A has full rank with probability 1.*

A proof of this can be can be found in [Feng and Zhang (2007)].

Lemma 2. *If A, B are two RIRE matrices of size (n_1, n_2) and (n_2, n_3) then their product $C = AB$ is of rank $\min(n_1, n_2, n_3)$.*

Proof. Since both A and B are RIRE matrices we know that rank(B) = $\min(n_3, n_2)$ and rank(A) = $\min(n_2, n_1)$ We thus want to prove that rank(C) = $\min(n_3, n_2, n_1)$. We can prove this by constructing each column of C one at a time. The i:th column c_i of C is given by $c_i := Ab_i$ where b_i denotes the i:th column of B.

We generate these vectors one at a time until either two things happen. Either the previously generated vectors $c_1, c_2, ..., c_i$ span the image space of A at which

point all subsequently generated vectors will lie in the span of the previously generated vectors, or $c_1, c_2, ..., c_i$ does not span the image space and thus only a subspace and the next vector c_{i+1} will thus with probability 1 not lie in the span (and therefore be linearly independent). In the second case the generation process keeps going until either the first scenario happens or we have generated all $c_1, c_2, ..., c_{n_1}$ vectors. If the first scenario occurs then the dimension of the image space of C is given by $\min(n_3, n_2)$ and in the second case it is given by n_1. Combining these results we get that the rank of the matrix C is given by $\min(n_3, n_2, n_1)$. □

We now state the theorem of interest and prove it. Equation (2.3) will then follows as a corollary.

Theorem 1. *If the matrix W is the product of $W_1, W_2, .., W_n$ RIRE matrices of size $(n_1, n_2), (n_2, n_3), ..., (n_n, n_{n+1})$ then the rank of W is given as*

$$\textit{rank}(W) = \min(n_1, n_2, ..., n_n, n_{n+1}).$$

Proof. We will prove this through induction. Lemma 2 proves the base case and we thus assume that it is true for a product of $n - 1$ RIRE matrices and want to prove that it is true for n RIRE matrices.

Given that we have

$$W = W_n W_{n-1} \cdots W_2 W_1 \tag{A.1}$$

and by denoting the $n - 1$ matrix product as $A := \tilde{W}_n \tilde{W}_{n-1} \cdots \tilde{W}_2$ we can proceed as in Lemma 2 and show that

$$\tilde{W}_R = A W_1 \tag{A.2}$$

is of rank $\min(n_{n+1}, n_n, ..., n_2, n_1)$. As before we can construct the columns of W sequentially as $Aw_{1,i}$. Either we generate enough vectors to span the image space of A or we generate n_1 linearly independent vectors. Since the dimension of the image space of A is now $\min(n_{n+1}, n_n, ..., n_3, n_2)$ the rank of $\tilde{W}_R$ is given by $\min(n_{n+1}, n_n, ..., n_3, n_2, n_1)$. □

Corollary 1. *At initialization, the rank of the matrix W_R in the local affine transformation for the neural network f_θ, $f_\theta(x) = W_R x + b_R,\ \forall x \in R$, is given as*

$$\textit{rank}(W_R) = \min(n_0, n_c, \min_i \textit{trace}(Z_i)).$$

Proof. As before we utilize the identity

$$W_R = W_n Z_{n-1} W_{n-1} \cdots Z_2 W_2 Z_1 W_1.$$

Since the effect of the diagonal matrices Z_i is only to zero out rows in the corresponding matrix W_i, we can form new RIRE matrices $\tilde{W}_i$ by removing the zero rows from $Z_i W_i$ and additionally setting $\tilde{W}_n := W_n$. Thus W_R is a product of the RIRE matrices $\tilde{W}_1, \tilde{W}_2, ..., \tilde{W}_n$ and since the rank of every RIRE matrix $\tilde{W}_i$ is given by $\min(trace(Z_i), n_i)$, the result follows. □

A.2 Experimental details

A.2.1 Details for Figure 2.1

We train a one layer fully connected network with 200 neurons in the hidden layer. The network is trained with stochastic gradient descent with a learning rate of 0.001, momentum of 0.8, batch size of 32 for 100 epochs.

A.2.2 Details for Figure 2.2

We train a fully connected two-layer network with 200 neurons in both hidden layers. We train with stochastic gradient descent with a momentum of 0.8 for 100 epochs with a batch size of 32 on FashionMNIST.

A.2.3 Details for Figure 2.3

The left plot was created by sampling a point in the input space and sampling points on concentric circles around the initial point until a point on a circle ends up in a different activation region. The singular values are then measured in both the initial region and the region associated with the point on the circle. This process is repeated 500 times for each neuron amount to get an estimate of the mean difference.

The right plot was created by sampling points in the input space and going a distance one away and checking if points a distance one away is in another activation region. If so then we measure the singular values in both the initial region and the region associated with the point a distance one away. If the point

a distance one away is not in a different region then a new point is sampled. This process is repeated 500 times to obtain an estimate of the mean difference.