On Noise Regularised Neural Networks: Initialisation, Learning and Inference

by

Arnu Pretorius

Dissertation presented for the degree of Doctor of Philosophy in the Faculty of Science at Stellenbosch University

Computer Science Division,
Department of Mathematical Sciences,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Promoters:

Dr R. S. Kroon Dr H. Kamper

December 2019

Stellenbosch University https://scholar.sun.ac.za

Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

This dissertation includes five original papers (two published and three under review). The development and writing of the papers were the principal responsibility of myself and, for each of the cases where this is not the case, a contribution statement is included in the dissertation indicating the nature and extent of the contributions of co-authors.

Date: December 2019

Copyright © 2019 Stellenbosch University
All rights reserved.

Stellenbosch University https://scholar.sun.ac.za

Abstract

On Noise Regularised Neural Networks: Initialisation, Learning and Inference

A. Pretorius

Computer Science Division,
Department of Mathematical Sciences,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Dissertation: PhD (Computer Science)

December 2019

Innovation in regularisation techniques for deep neural networks has been a key factor in the rising success of deep learning. However, there is often limited guidance from theory in the development of these techniques and our understanding of the functioning of various successful regularisation techniques remains impoverished.

In this work, we seek to contribute to an improved understanding of regularisation in deep learning. We specifically focus on a particular approach to regularisation that injects noise into a neural network. An example of such a technique which is often used is *dropout* (Srivastava *et al.*, 2014).

Our contributions in noise regularisation span three key areas of modeling: (1) learning, (2) initialisation and (3) inference. We first analyse the **learning** dynamics of a simple class of shallow noise regularised neural networks called *denoising autoencoders* (DAEs) (Vincent *et al.*, 2008), to gain an improved understanding of how noise affects the learning process. In this first part, we observe a dependence of learning behaviour on initialisation, which leads us to study how noise interacts with the **initialisation** of a *deep* neural network in terms of signal propagation dynamics during the forward and backward pass. Finally, we consider how noise affects **inference** in a Bayesian context. We mainly focus on fully-connected feedforward neural networks with rectifier linear unit (ReLU) activation functions throughout this study.

To analyse the learning dynamics of DAEs, we derive closed form solutions to a system of decoupled differential equations that describe the change in scalar weights during the course of training as they approach the eigenvalues of the input covariance matrix (under a convenient change of basis). In terms of initialisation, we use mean field theory to approximate the distribution of the pre-activations of individual neurons, and use this to derive recursive equations that characterise the signal propagation behaviour of the noise regularised network during the first forward and backward pass of training. Using these equations, we derive new initialisation schemes for noise regularised neural networks that ensure stable signal propagation. Since this analysis is only valid at initialisation, we next conduct a large-scale controlled experiment, training thousands of networks under

Stellenbosch University https://scholar.sun.ac.za

a theoretically guided experimental design, for further testing the effects of initialisation on training speed and generalisation. To shed light on the influence of noise on inference, we develop a connection between randomly initialised deep noise regularised neural networks and Gaussian processes (GPs)—non-parametric models that perform exact Bayesian inference—and establish new connections between a particular initialisation of such a network and the behaviour of its corresponding GP. Our work ends with an application of signal propagation theory to approximate Bayesian inference in deep learning where we develop a new technique that uses *self-stabilising priors* for training deep Bayesian neural networks (BNNs).

Our core findings are as follows: noise regularisation helps a model to focus on the more prominent statistical regularities in the training data distribution during learning which should be useful for later generalisation. However, if the network is deep and not properly initialised, noise can push network signal propagation dynamics into regimes of poor stability. We correct this behaviour with proper "noise-aware" weight initialisation. Despite this, noise also limits the depth to which networks are able to train successfully, and networks that do not exceed this depth limit demonstrate a surprising insensitivity to initialisation with regards to training speed and generalisation. In terms of inference, noisy neural network GPs perform best when their kernel parameters correspond to the new initialisation derived for noise regularised networks, and increasing the amount of injected noise leads to more constrained (simple) models with larger uncertainty (away from the training data). Lastly, we find our new technique that uses self-stabilising priors makes training deep BNNs more robust and leads to improved performance when compared to other state-of-the-art approaches.

Stellenbosch University https://scholar.sun.ac.za

Opsomming

Ruisgeregulariseerde Neurale Netwerke: Inisialisering, Leer en Inferensie

A. Pretorius

Rekenaarwetenskap Afdeling,
Departement van Wiskundige Wetenskappe,
Universiteit van Stellenbosch,
Privaatsak X1, Matieland 7602, Suid Afrika.

Proefskrif: PhD (Rekenaarwetenskap)

Desember 2019

Innovasie in regulariseringstegnieke vir diep neurale netwerke is 'n belangrike aspek van die toenemende sukses van diepleer. Daar is egter beperkte teoretiese leiding in die ontwikkeling van hierdie tegnieke, en ons begrip van hoe verskeie suksesvolle regulariseringstegnieke funksioneer is steeds onvolledig.

Hierdie proefskrif poog om 'n bydra te maak tot 'n beter begrip van regularisering in diepleer. Ons fokus spesifiek op benaderings wat van ruis gebruik maak om neurale netwerke te regulariseer. Een voorbeeld van so 'n tegniek wat gereeld gebruik word, is weglating ("dropout") (Srivastava *et al.*, 2014).

Ons bydrae tot ruisregularisering span drie sleutelareas van modellering: (1) leer, (2) inisialisering en (3) inferensie. Ons ondersoek eers die leerdinamika van 'n eenvoudige klas van vlak ruisgeregulariseerde neurale netwerke, genaamd ontruisende outo-enkodeerders (DAEs) (Vincent *et al.*, 2008), om 'n beter begrip te kry van hoe ruis die leerproses beïnvloed. In die eerste deel, neem ons waar dat leergedrag afhanklik is van inisialisering. Dit motiveer 'n studie waar ons die wisselwerking tussen ruis en die inisialisering van diep neurale netwerke bestudeer in terme van die dinamika van seinvloei gedurende die aanvanklike voorwaartse en terugwaartse deurvloei. Laastens word daar gekyk na die invloed van ruis in die Bayes-konteks. Hierdie studie fokus hoofsaaklik op volledig-gekoppelde vorentoevoer neurale netwerke met die gerektifiseerde lineêre eenheid (ReLU) aktiveringsfunksie.

Om die leerdinamika van DAEs te ontleed, lei ons geslotevorm oplossings af vir 'n stelsel ontkoppelde differensiaalvergelykings wat die verandering in skalaargewigte tydens die afrigproses beskryf, soos wat hulle na die eiewaardes van die toevoerkovariansiematriks neig (onder 'n gerieflike basisverandering). Wat inisialisering betref, gebruik ons gemiddelde-veldteorie om die verdeling van die voor-aktiverings van individuele neurone te benader, en gebruik ons dan hierdie resultate om rekursiewe vergelykings af te lei wat die seinvloei gedrag van ruisgeregulariseerde netwerke tydens die eerste voorwaartse en terugwaartse deurvloei beskryf. Ons gebruik hierdie vergelykings om nuwe inisialiseringskemas vir ruisgeregulariseerde neurale netwerke wat stabiele seinvloei verseker te

Stellenbosch University https://scholar.sun.ac.za

verkry. Aangesien hierdie analise slegs geldig is tydens inisialisering, voer ons 'n grootskaalse gekontroleerde eksperiment uit deur duisende netwerke af te rig volgens 'n geskikte eksperimentele ontwerp, om sodoende die effek van inisialisering op die afrigspoed en veralgemening van die afrigte netwerke te toets. Om lig te werp op die effek van ruis op inferensie, ontwikkel ons 'n verband tussen ewekansigge geïnitialiseerde diep ruisgeregulariseerde neurale netwerke en Gaussiese prosesse (GPs)—nie-parametriese modelle wat presiese Bayesiaanse inferensie uitvoer—sowel as nuwe verbande tussen 'n spesifieke inisialisering van die netwerk en die optrede van die ooreenstemmende GP. Die proefstuk word afgesluit met 'n toepassing van seinvloeiteorie op Bayesiaanse inferensie in diep neurale netwerke, waar ons 'n nuwe tegniek, wat gebruik maak van "*selfstabiliserende priors*", ontwikkel.

Ons kernbevindinge is as volg: ruisregularisering help 'n model om te fokus op die meer prominente statistiese reëlmatighede in die verdeling van die afrigdata, wat vir latere veralgemening nuttig behoort te wees. As die netwerk egter diep is en nie behoorlik geïnisialiseer is nie, kan ruis veroorsaak dat die dinamika van die netwerk se seinvloei onstabiel raak. Ons kan egter hierdie optrede teenwerk met behoorlike "ruisbewuste" gewigsinisialisering. Nietemin, beperk ruis die diepte waartoe netwerke suksesvol afgerig kan word, en netwerke wat nie hierdie dieptebeperking oorskry nie, toon 'n verbasende onsensitiwiteit tot inisialisering met betrekking tot hul afrigspoed en veralgemening. Wat inferensie betref, presteer ruisige neurale netwerk GPs die beste wanneer hul kernparameters ooreenstem met die nuwe inisialisering wat afgelei is vir ruisgeregulariseerde netwerke, en 'n toename in die hoeveelheid ruis wat bygevoeg word lei tot meer beperkte (eenvoudige) modelle met groter onsekerheid (weg van die afrigtingsdata). Laastens vind ons dat ons nuwe tegniek, wat gebruik maak van "selfstabiliserende priors", die afrigting van Bayesiaanse neurale netwerke meer robuust maak en tot verbeterde prestasie lei in vergelyking met ander moderne benaderings.

Stellenbosch University https://scholar.sun.ac.za

Acknowledgements

I would like to start by deeply thanking both my supervisors Dr Steve Kroon and Dr Herman Kamper. Steve, thank you for being so meticulous in your approach, for always making the time to see me when I need your guidance or advice, and for being a wonderful advisor. Herman, thank you for all your advice, support, encouragement and all the numerous small discussions that made a world of difference (even if it did not feel to you that way) for my confidence as a researcher. I have truly learned a lot from both of you.

I am extremely thankful for all my lab mates at the MIH Media Lab at Stellenbosch University. You know who you are! Thanks for all the great conversations, ideas, games, advice, tea, milk and just for being good company during countless late nights and early mornings. Especially, Armand du Plessis, I'm going to miss our early morning philosophical discussions over a cup of tea.

I would also like to express my deepest thanks and gratitude to all the students who have worked with me on some of the research projects in this dissertation, either as participants in initial coding sprints or as full research collaborators: Elan van Biljon, Felix McGregor, Benji van Niekerk, Ryan Eloff, Steve James and Matthew Reynard. It was truly a pleasure to work with all of you. Special thanks to Elan van Biljon. You are a great friend and all-round wonderful person to work with.

I would like to thank Dr McElory Hoffmann and Dirko Coetsee from Praelexis for letting me do some of my research in their workspace and for stimulating conversations throughout this time. Also, a special thanks to Dr Bubacarr Bah for inviting me to give talks on my research at the Southern African Mathematical Science Association (SAMSA) Conference and at the African Institute for Mathematical Sciences (AIMS) in South Africa.

I would also like to thank Dr Benjamin Rosman for inviting me to give a talk at the RAIL lab at Wits University. The RAIL lab is home to truly some of the most impressive researchers and students on the African continent. Your high and consistent achievements have really inspired me to always try my best and reach for ambitious goals.

The Deep Learning Indaba (DLI), the largest annual meeting of the African machine learning community, has shaped my trajectory as a young aspiring researcher enourmously. I would therefore like to deeply thank the following people who are behind this great initiative to strengthen machine learning in Africa: Shakir Mohamed, Ulrich Paquet, Avishkar Bhoopchand, Stephan Gouws, Benji Rosman, Willie Brink, Vukosi Marivate, Nyalleng Moorosi, Richard Klein, Muthoni Wanyoike, Kathleen Siminyu, Daniela Massiceti and Herman Kamper. The impact the DLI has had over the years, not just on me, but on the entire international community, is truly amazing. We as a community cannot thank you enough.

I am especially indebted to Ulrich Paquet, who has been a wonderful mentor and a friend. Thank you for your support and advice, letting me stay with you during my visit to the UK and introducing me to the Computational and Biological Learning (CBL) lab

Stellenbosch University https://scholar.sun.ac.za

at Cambridge University.

I would like to thank the following funding sources who have supported me at various stages during my research: the Centre for Artifical Intelligence Research (CAIR) in South Africa, the Harry Crossley foundation, Google, Nvidia, NeurIPS and ICML.

Second to last, a huge thank you to all my family and friends! Specifically my parents, Otto Pretorius and Ina Pretorius, thank you for your tremendous encouragement and support throughout my graduate years. And my brothers Ruan Pretorius and Daru Pretorius, for always being there for me when I need you, even though you have no idea what I'm actually doing half of the time.

Lastly, I would like to thank my wife, Jani Pretorius. From the bottom of my heart, thank you for all your love, kindness, understanding and support. Without you none of this would have been possible.

Stellenbosch University https://scholar.sun.ac.za

Contents

Stellenbosch University https://scholar.sun.ac.za

Stellenbosch University https://scholar.sun.ac.za

Chapter 1

Introduction

Regularisation plays a central role in deep learning. However, several state-of-the-art innovations in regularisation are still only heuristically motivated, or have largely come about through experimentation. As a result, we still lack a solid understanding of how and why different regularisation strategies work, and in what ways these strategies affect the different stages of modeling. Furthermore, a better understanding of regularisation for deep learning might enable a more principled approach to neural network design and regularisation.

The aim of this thesis is to contribute towards an improved understanding of regularisation for deep learning. We focus on a particular regularisation strategy that injects noise into a neural network during training. This strategy has been shown to be especially effective for a wide range of different deep learning models and applications (Sietsma and Dow, 1991; Vincent *et al.*, 2008; Srivastava *et al.*, 2014).

The outline of this thesis is as follows: the remainder of this introductory chapter will provide general background relevant for contextualising the contributions in this thesis. Chapters 2 to 6 form the main content of the thesis. Each chapter is based on a paper and consists of an introduction, a contribution statement, the paper itself and a discussion. We also provide background material separately in each chapter as required. Finally, we conclude in Chapter 7 with a summary of the thesis and suggested future work.

1.1 Supervised machine learning

Statistical machine learning aims to create intelligent computer systems capable of making appropriate decisions in a variety of complex situations. Typically, these computer systems acquire their decision-making ability through *machine learning algorithms*. By combining data collected from the real world with an assumed statistical model describing the relationships between examples, machine learning algorithms learn how to make decisions through error correction by optimising a pre-specified *objective function*.

More specifically, it is possible to implement a wide variety of well-known supervised machine learning algorithms using the following generic template: given a data set of independent and identically distributed (i.i.d.) input-output pairs $\mathcal{D} = \{(\mathbf{x}_i, y_i), i = 1, ..., N\}$, we assume a probability distribution for y given $\mathbf{x}$ of the form $y|\mathbf{x} \sim p(\alpha)$, where α represents a set of parameters describing the distribution (e.g. μ and σ^2, the mean and variance describing a Gaussian distribution). To model the expected relationship $\mathbb{E}[y|\mathbf{x}]$, we use a parameterised function of the input $f(\mathbf{x}, \theta)$, where the adjustable parameters

Stellenbosch University https://scholar.sun.ac.za

θ are responsible for encoding the mapping from the inputs to the (expected) outputs. Finally, we estimate suitable parameters θ^* by maximising the likelihood of the parameters given the observed data, or equivalently, minimising the negative log-likelihood

$$\mathcal{L}(\mathcal{D}, \theta) = \sum_{i=1}^{N} -\log p(y_i|\mathbf{x}_i, \theta). \tag{1.1.1}$$

Many (but not all) supervised machine learning algorithms can be considered as special cases of the above general construction. A powerful alternative modeling paradigm is Bayesian machine learning, which we discuss in Chapters 5 and 6.

These machine learning algorithms aim to uncover statistical regularities in the data that describe the underlying data generating mechanism, and encode its typical behaviour in the model parameters during learning. This task is far from simple, and tractable approaches often rely on strong prior beliefs that are assumed to resonate with natural phenomena of interest.

For example, using the above generic template, if we assume a Gaussian conditional distribution $y|\mathbf{x} \sim \mathcal{N}(f(\mathbf{x}, \theta), \sigma^2)$, with $f(\mathbf{x}, \theta = \{\mathbf{w}, b\}) = \mathbf{w} \cdot \mathbf{x} + b$, we obtain the algorithm for *linear regression*, with squared-error loss: $\mathcal{L}(\mathcal{D}, \theta) \propto \sum_{i=1}^{N}(f(\mathbf{x}_i, \theta) - y_i)^2$. Here, $\mathbf{w}$ and b are the adjustable parameters of the model, referred to as the weights and the bias respectively. Similarly, if we assume $y|\mathbf{x}$ is Bernoulli distributed with parameter $f(\mathbf{x}, \theta) = \xi(\mathbf{w} \cdot \mathbf{x} + b)$, where ξ is the logistic function, we obtain *logistic regression*. These two algorithms in particular have proved to be very useful for a wide range of different regression and classification tasks. However, at their core is the assumption of *local smoothness* in the input space, i.e. $f(\mathbf{x}) \approx f(\mathbf{x} + \epsilon)$, for some small perturbation vector ϵ. If observed data densely populate the input space, local smoothness is a reasonable assumption and learned relationships generally extrapolate well to other unobserved areas of the space.

Unfortunately, many real-world phenomena are embedded in extremely high dimensional spaces. This leads to poor sample efficiency, requiring exponentially more data per dimension to maintain the same level of local density. For a fixed number of data points and a growing number of input dimensions, what is considered local in terms of the nearest neighbours to a point in the input space, can no longer be considered local in terms of the volume of space required to encapsulate all these neighbouring points. This is often referred to as the *curse of dimensionality* (Bellman, 1961), and many traditional machine learning algorithms suffer from it (including popular kernel and tree-based methods) (Hastie *et al.*, 2009).

One approach to overcoming the curse of dimensionality is to disentangle the factors of variation in the input representation by transforming the factors into *distributed representations* (Goodfellow *et al.*, 2016). To see why this can be effective, consider a data generator that produces images of cats and dogs, that can either be black or white in colour. Instead of having representations encoding each different class e.g. *blackCat, whiteCat, blackDog, whiteDog*, sometimes referred to as *localist* representations, distributed representations disentangle the factors of variation into colour {*black, white*} and animal {*cat, dog*} and can construct all possible combinations as the Cartesian product of the two sets. More generally, given an input vector $\mathbf{x}$ of size d, with binary encoding at each index, a unique localist representation for each configuration would require collecting $O(2^d)$ data points to have knowledge of all possible configurations. In contrast, using a distributed representation would only require $O(d)$ data points to obtain the same level of knowledge.

Stellenbosch University https://scholar.sun.ac.za

Distributed representations can clearly be of great benefit. However, disentangling the factors of variation in extremely complex observed phenomena can be very difficult. Hand engineered preprocessing techniques, or *feature engineering*, is one approach, but usually requires expert domain knowledge which limits its applicability in domains that are still poorly understood, e.g. language development in infants (Räsänen, 2012; Räsänen and Rasilo, 2015). An alternative approach is to instead *learn* distributed representations.

1.2 Deep learning

Deep learning attempts to learn distributed representations as part of a more general purpose approach to statistical machine learning. The primary assumptions underlying deep learning are: (1) most observed data stem from data generating mechanisms that are *compositional* in nature; and (2) that this hierarchical composition consists of independent factors of variations at different levels of abstraction (i.e. representations are distributed across both basic low-level concepts and higher-level complex or more abstract concepts) (Goodfellow *et al.*, 2016). Deep learning still follows the same design pattern as the generic template presented earlier for constructing different machine learning algorithms. It also relies on the local smoothness assumption. But what is important, is that for many tasks of interest the additional assumptions of (1) and (2) above tend to hold, and deep learning is able to *overcome the curse of dimensionality by combining distributed representations with composition.*

For example, a deep learning approach to modeling a continuous output variable y, might follow the same recipe as linear regression, except that now

$$f(\mathbf{x}, \theta) = \mathbf{w}^{out} \cdot \mathbf{x}^L + b^{out}, \tag{1.2.1}$$

where the original input $\mathbf{x}$ is first transformed into $\mathbf{x}^L$ using a multi-layered *deep neural network*:

$$\mathbf{x}^l = \phi(\mathbf{h}^l), \qquad \mathbf{h}^l = W^l \mathbf{x}^{l-1} + \mathbf{b}^l, \qquad \text{for } l = 1, ..., L \tag{1.2.2}$$

with input $\mathbf{x}^0 = \mathbf{x}$. Here ϕ is an element-wise *activation function*, the W^l are the weight matrices, and $\mathbf{b}^l$ are the bias vectors. We refer to $\mathbf{h}^l$ as the vector of hidden unit *pre-activations* and to $\mathbf{x}^l$ as the vector of hidden unit *activations* associated with a specific layer l. The above network is an example of an L-hidden-layer *fully-connected feedforward* neural network with a linear output layer. The terms "fully-connected" and "feedforward" refer to the *architecture* of the network. Different architectures attempt to create different structural priors that are well suited for a particular task. Other popular architectures include, *convolutional* architectures (LeCun, 1989) for spatial data and *recurrent* architectures (Rumelhart *et al.*, 1988; Hochreiter and Schmidhuber, 1997) for temporal data.

The network $f(\mathbf{x}, \theta)$ is parameterised by $\theta = \{\mathbf{w}^{out}, b^{out}, \mathbf{b}^1, ..., \mathbf{b}^L, W^1, ..., W^L\}$, where at each layer of the network the inputs are transformed by the weights W^l and bias vectors $\mathbf{b}^l$ and then passed through an activation functions ϕ. By learning through multiple stacked layers in this way, the network is able to encode into the parameters suitable transformations required to create hierarchically composed distributed representations of the inputs. These representations then make downstream regression or classification tasks more tractable when dealing with complex high-dimensional data.

Deep learning has made remarkable progress on long-standing and fundamental challenges in machine learning and artificial intelligence. It has been hugely successful for

Stellenbosch University https://scholar.sun.ac.za

example in producing models that are able to recognise objects (Krizhevsky *et al.*, 2012; Szegedy *et al.*, 2013; Erhan *et al.*, 2014) and speech (Hinton *et al.*, 2012; Graves and Jaitly, 2014), generate comprehensible text (Sutskever *et al.*, 2011; Graves, 2013), and translate sentences (Cho *et al.*, 2014; Sutskever *et al.*, 2014).

Although the idea of representation learning was the driving force behind the development of deep learning, these algorithms still rely heavily on *regularisation* and clever *optimisation*, to be able to perform well. One way to think of deep learning is in terms of search, where algorithms run the following procedure: (1) define a *hypothesis space* containing different possible hypotheses explaining the underlying data generating mechanism and (2) search this space to find the hypothesis (a mapping from learned representations to output) that best describes the general trends in the observed data. By respectively playing key roles in steps (1) and (2) above, regularisation and optimisation form the pillars of deep learning. In this thesis, we focus on regularisation.

1.3 Regularisation

Machine learning differs from pure optimisation in the sense that the task is to learn from a finite collection of observed data (i.e. to optimise over some training set), but then *generalise* to the underlying data generator of (possibly infinitely many) previously unseen test observations. We assume that this is possible as long as the observed training data and the test data come from the same distribution. If the samples that comprise the training and test sets are i.i.d., the expected performance of a model for a given set of parameters will be equal on both sets. However, during optimisation the model parameters are specifically modified using information from the training set. If the model starts to incorporate idiosyncrasies specific to the training set, it will not generalise well to the true underlying distribution. This generally happens when the *complexity* of the model is inappropriate for the task at hand.

The complexity of a model refers to the variety of functions it is able to express. These functions can range from being too simple to fit the data (underfitting), to being unnecessarily complex (overfitting). For a fixed training set, an appropriate level of complexity typically lies somewhere between these two extremes, as illustrated in Figure 1.1. A key result from *statistical learning theory* is that there exists an upper bound on the difference between training and generalisation performance. This bound grows with model complexity, but shrinks with dataset size (Vapnik, 1995).

Solutions in the hypothesis space will be associated with different levels of complexity. When searching the hypothesis space, pure optimisation on the training set will favour models of greater complexity, since these will correspond to the functions most capable of minimising the training loss. Unfortunately this does not guarantee that the test loss will also be low. On the contrary, the most common outcome in this situation is that the model overfits and performs poorly on new test data. Therefore, pure optimisation is not sufficient for developing useful machine learning algorithms. What we require in addition to optimisation is effective regularisation.

Regularisation can broadly be defined as follows:

> **Definition 1: Regularisation** *refers to any strategy that modifies or encodes preferences into the hypothesis space being searched during optimisation, with the single aim of improving generalisation.*

Stellenbosch University https://scholar.sun.ac.za

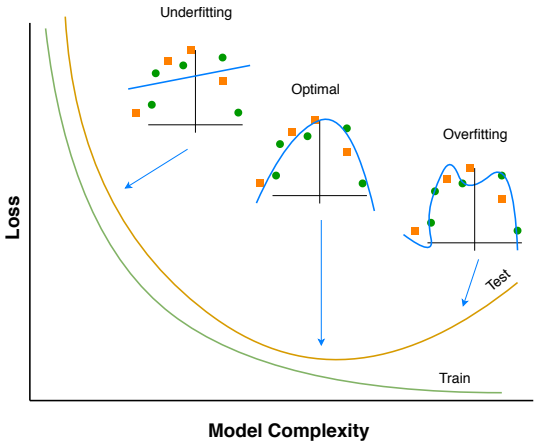


Figure 1.1: *Model complexity, underfitting and overfitting*: The plot shows typical training and test loss curves as a function of model complexity. Green circles represent training points and orange squares represent test points, with fitted regression functions shown in blue.

Simply put, regularisation aims to ensure that optimisation on the training set is more likely to translate into good performance on the test set. This does not mean that regularisation improves optimisation. In fact, the opposite is often the case, with added regularisation leading to more difficult optimisation problems and reduced performance on the training set. Instead, regularisation only serves as a way to build preferences into the hypothesis space that guide the search procedure. This guidance is provided irrespective of the difficulty in optimisation associated with each preference.

Regularisation can either be implicit, for example designing algorithms based on the assumptions of smoothness and hierarchical composition. Alternatively, it can be more explicit, such as directly limiting the hypothesis space or constraining the optimisation procedure.

In this thesis, we focus on an explicit form of regularisation that injects noise into a neural network during training. Specifically, for an input $\mathbf{x}^l$ at an arbitrary layer $l = 0, 1, ..., L$, in a deep neural network, we consider general noise injection of the form $\tilde{\mathbf{x}}^l = \mathbf{x}^l \odot \epsilon^l$. Here, $\odot$ denotes element-wise addition or multiplication and ϵ is a noise vector sampled from a pre-specified noise distribution. We refer to this specific form of regularisation simply as *noise regularisation*. We consider in our investigations both additive and multiplicative noise sampled from different noise distributions and injected at different layers of the network.

1.4 Thesis contributions

Our contributions focus on understanding the effect of noise regularisation on different stages of the modeling process, namely at initialisation (the setting of the starting parameter values), during learning (optimisation) and when performing inference (prediction and uncertainty estimation). See Figure 1.2 for a visual overview.

In each chapter **we make the following contributions**:

- **Chapter 2** investigates the learning dynamics of linear autoencoder neural networks with input noise injection. We derive new closed-form solutions to the learning dynamics of these models in expectation over the sampled noise. We find that noise helps the model to avoid learning low variance directions in the input space, while leaving higher variance directions largely unaffected. Our findings also highlight

Stellenbosch University https://scholar.sun.ac.za

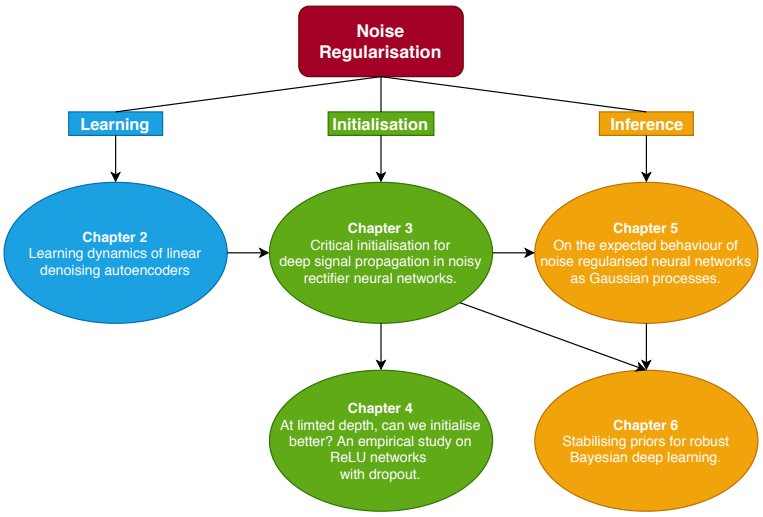


Figure 1.2: *Contributions: outline, flow and dependencies.* Our contributions fall into three broad categories: learning (blue), initialisation (green) and inference (orange). Black arrows show dependencies between papers. For example, Chapter 4 builds on work presented in Chapter 3, and Chapter 6 was influenced by the ideas in Chapters 3 and 5.

the effects of noise from different initialisations, as well as provide connections with other well-known regularisation techniques such as weight decay and early stopping. Specifically, noise regularisation is shown to have a similar regularisation effect as weight decay, but with faster training dynamics.

> **Paper**: Pretorius, A., Kroon, S., & Kamper, H. Learning dynamics of linear denoising autoencoders. *International Conference on Machine Learning (ICML, 2018).*

- **Chapter 3** moves to the nonlinear case and focuses specifically on initialisation for noise regularised neural networks. We investigate how noise interacts with initialisation during the forward pass in deep fully-connected feedforward neural networks that use Rectified Linear Unit (ReLU) activation functions. We derive a new critical initialisation for noise regularised neural networks that is optimal in terms of specific criteria for stable forward signal propagation. Our initialisation also helps to shed light on previously unexplained phenomena, for example why a dropout rate of 0.5 often works well in practice when combined with a well-known initialisation strategy. Finally, we show how noise regularisation limits the maximum depth to which a ReLU neural network can be trained.

  > **Paper**: Pretorius, A., van Biljon, E., Kroon, S. & Kamper, H. Critical initialisation for deep signal propagation in noisy rectifier neural networks. *Advances in Neural Information Processing Systems (NeurIPS, 2018).*

- **Chapter 4** directly builds on the work in Chapter 3 and investigates the following question: if critical initialisation is only a requirement for *deep* signal propagation, and noise *limits training depth*, are there perhaps alternative (non-critical) initialisations that perform better for noisy networks within this depth limit? We perform a large-scale controlled experiment by training over $12,000$ networks to compare critical versus non-critical initialisations. A statistical analysis of our results shows that for a wide range of initialisations around criticality, there is no statistically significant difference in training speed or generalisation when compared to the critical initialisation. Our analysis seems to indicate that neural networks of shallow to moderate depth are largely insensitive to initialisation.

Stellenbosch University https://scholar.sun.ac.za

> **Paper**: Pretorius, A., van Biljon, E., van Niekerk, B., Eloff, R., Reynard, M., James, S., Rosman, B., Kamper, H., & Kroon, S. At limted depth, can we initialise better? An empirical study on ReLU networks with dropout. *Preprint (arXiv, 2019).*

- **Chapter 5** connects noise regularised deep neural networks with Gaussian processes (GPs). Prior work has shown that a deep neural network at initialisation is equivalent in the large width limit to a GP. The resulting GP's kernel parameters depend on the network depth and the scale of the weights and biases. We extend these results to include noise regularisation. We find that the best performing GPs are those that have their kernel parameters set to correspond to the values specified by the initialisation derived in Chapter 3. Furthermore, we show how noise regularisation in a neural network affects the covariance of the corresponding noisy GP. As more noise is injected into the network, the covariance tends to become diagonal implying complete independence between training points. This leads to a stronger prior for simple functions and larger uncertainty in the posterior predictive distribution when performing exact Bayesian inference.

  > **Paper**: Pretorius, A., Kamper, H., & Kroon, S. On the expected behaviour of noise regularised neural networks as Gaussian processes. *Preprint (arXiv, 2019).*

- **Chapter 6** considers approximate Bayesian inference in deep Bayesian neural networks (BNNs). Inspired by the analysis techniques in Chapters 3 and 5, we develop *self-stabilising priors*, an adaptive Monte Carlo variational inference method for BNNs. This approach is able to successfully train much deeper BNNs in more noisy settings (with large uncertainty in the weights) than other state-of-the-art methods.

  > **Paper**: McGregor, F., Pretorius, A., du Preez, J., & Kroon, S. Stabilising priors for robust Bayesian deep learning. *Advances in Neural Information Processing Systems workshop on Bayesian Deep Learning (NeurIPS BDL workshop, 2019).*

Finally, in **Chapter 7** we provide a summary of the work presented in the thesis along with concluding remarks and suggestions for future work.

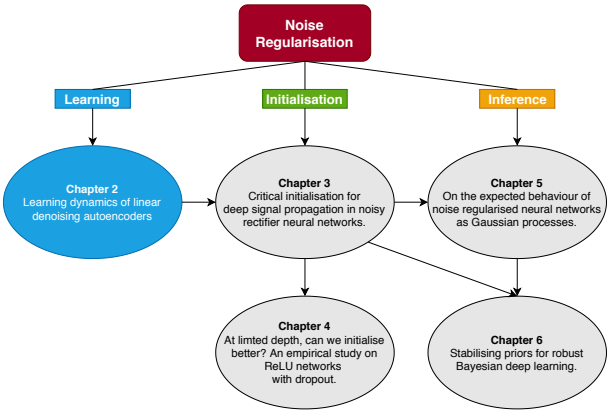
Stellenbosch University https://scholar.sun.ac.za

Chapter 2

Learning dynamics of denoising autoencoders

Paper
Pretorius, A., Kroon, S., & Kamper, H. Learning dynamics of linear denoising autoencoders. *International Conference on Machine Learning (ICML, 2018)*.
Available at: http://proceedings.mlr.press/v80/pretorius18a/pretorius18a.pdf

Code
Pretorius, A. (2018). Code for: Learning dynamics of linear denoising autoencoders.
Available at: https://github.com/arnupretorius/lindaedynamics_icml2018.



2.1 Introduction

As mentioned in Chapter 1, the success of deep learning is built upon a modeling approach that seeks to *learn*, instead of hand engineer, useful features for a particular task. Autoencoder (AE) neural networks are especially useful in this regard. AEs are frequently used as *feature extractors*. A good feature extractor network learns useful latent feature representations which are "extracted" from the internal layers of the network for use in subsequent tasks.

During feature extraction, noise regularisation forces an AE network to focus on the more generalisable statistical aspects of the input distribution. By ignoring less relevant information (hopefully specific to the training set) a type of noise regularised AE called a *denoising autoencoder* (DAE), which we consider in this chapter, generally learns more

Stellenbosch University https://scholar.sun.ac.za

useful latent feature representations than an unregularised AE (Vincent *et al.*, 2008). However, even though DAEs are empirically very useful, our theoretical understanding of how noise affects the learning of DAEs remains limited.

In this chapter, we investigate the dynamics of learning in DAEs to uncover how the directions of variation in the input distribution are learned over the course of training. Our approach largely follows that of Saxe *et al.* (2014). Specifically, we focus on the learning dynamics of *linear* DAEs in expectation over the noise distribution. Although the models we study are linear, their *learning dynamics are nonlinear* and closely resemble the learning dynamics empirically observed in nonlinear networks. We make use of a convenient change of basis that aligns the weights with the directions of variation in the input covariance. This allows the dynamics of learning to be described in terms of individual scalar dynamics for each eigenvalue associated with a specific direction. We derive exact solutions to differential equations describing the change in the map approximating each eigenvalue under the assumption of a small enough learning rate for continuous gradient descent dynamics. We find that the noise in DAEs helps the network to focus on learning only the larger variance directions during training, while ignoring smaller variance directions. We compare these dynamics with those for weight decay, a popular form of regularisation that adds a weight norm penalty to the loss function (Krogh and Hertz, 1992). Our analysis shows that the noise regularisation of DAEs has a similar regularisation effect to weight decay, but with faster training dynamics. Lastly, we verify that our theoretical predictions approximate learning dynamics on real-world data and qualitatively match observed dynamics in nonlinear DAEs.

2.2 Background

Autoencoding neural networks (LeCun, 1987; Bourlard and Kamp, 1988; Hinton and Zemel, 1994) attempt to learn an input data representation by modeling the inputs as the expected outputs. Although autoencoding is a form of *unsupervised learning* (learning without output labels), autoencoders generally follow the same generic template for supervised machine learning algorithms as outlined in Chapter 1. Specifically, we assume a distribution for the inputs conditioned on the inputs themselves and model the expected value of this distribution using a parameterised function. To find the optimal parameters, we minimise the negative log-likelihood.

The motivation behind autoencoder networks is as follows: if a layered network is able to accurately predict its own input, it is likely that the representations in its intermediate layers capture important characteristics of the input distribution. If this is the case, we can extract these representations from the network and use them in various downstream tasks. For this to be possible, the network model must be complex enough to capture important variations in the input, but at same time, not be so complex that it can simply represent the identity mapping. Therefore, for autoencoders to be useful, it is crucial that they incorporate some form of regularisation.

An effective way to regularise autoencoders is to inject noise into their inputs (Vincent *et al.*, 2008). This type of regularised autoencoder is known as a *denoising autoencoder* (DAE). For example, we can consider a real-valued input $\mathbf{x}$ injected with additive isotropic Gaussian noise, producing a corrupted version $\tilde{\mathbf{x}} = \mathbf{x} + \epsilon$, where $\epsilon \sim \mathcal{N}(\mathbf{0}, s^2 I)$. By assuming the following conditional distribution over the inputs, $\mathbf{x}|\tilde{\mathbf{x}} \sim \mathcal{N}(f(\tilde{\mathbf{x}}, \theta), \Sigma)$, we

Stellenbosch University https://scholar.sun.ac.za

model $\mathbb{E}[\mathbf{x}|\tilde{\mathbf{x}}]$ using a neural network

$$f(\tilde{\mathbf{x}}, \theta) = W^{out}\mathbf{x}^L + \mathbf{b}^{out},$$

where

$$\mathbf{x}^l = \phi(\mathbf{h}^l), \qquad \mathbf{h}^l = W^l\mathbf{x}^{l-1} + \mathbf{b}^l, \qquad \text{for } l = 1, ..., L \tag{2.2.1}$$

with $\mathbf{x}^0 = \tilde{\mathbf{x}}$. When $L = 1$, the internal representations of the input data get extracted from a single intermediate hidden layer, i.e. $\mathbf{h}^1$.

An implicit assumption often useful when designing machine learning algorithms is that almost all of the structure observed in the natural world resides on some lower-dimensional manifold. As an example, consider image data. The number of pixel configurations that produce meaningful images of real-world objects is far less than the total number of possible image configurations. Similar observations apply to many other different data types, e.g. speech or text data (Silberer and Lapata, 2012; Kamper *et al.*, 2015). This means that we can expect the real-world input distributions we actually observe to often be of a far smaller intrinsic dimension.

DAEs have been interpreted as a technique that pushes data off of its intrinsic manifold using noise injection, only to learn how to map the data back onto this manifold (Vincent *et al.*, 2008). Reconstructing the original inputs in this way forces DAEs to learn a type of (local) coordinate system. The axes of this coordinate system then span the directions of variation in the input space that are useful for getting back onto the manifold. Importantly, the directions orthogonal to this span, which are less useful for reconstruction, get ignored in the learned representations.

2.3 Contribution statement

The idea for this work was inspired by discussions with Andrew Saxe at the 2016 PRASA-Robmech conference in Stellenbosch, South Africa. I derived the theoretical results, performed the experiments and wrote the paper. Dr Steve Kroon provided technical feedback. Dr Herman Kamper provided general feedback and useful editorial suggestions.

Stellenbosch University https://scholar.sun.ac.za

Learning Dynamics of Linear Denoising Autoencoders

Arnu Pretorius [1 2] **Steve Kroon** [1 2] **Herman Kamper** [3]

Abstract

Denoising autoencoders (DAEs) have proven useful for unsupervised representation learning, but a thorough theoretical understanding is still lacking of how the input noise influences learning. Here we develop theory for how noise influences learning in DAEs. By focusing on linear DAEs, we are able to derive analytic expressions that exactly describe their learning dynamics. We verify our theoretical predictions with simulations as well as experiments on MNIST and CIFAR-10. The theory illustrates how, when tuned correctly, noise allows DAEs to ignore low variance directions in the inputs while learning to reconstruct them. Furthermore, in a comparison of the learning dynamics of DAEs to standard regularised autoencoders, we show that noise has a similar regularisation effect to weight decay, but with faster training dynamics. We also show that our theoretical predictions approximate learning dynamics on real-world data and qualitatively match observed dynamics in nonlinear DAEs.[*]

1. Introduction

The goal of unsupervised learning is to uncover hidden structure in unlabelled data, often in the form of latent feature representations. One popular type of model, an autoencoder, does this by trying to reconstruct its input (Bengio et al., 2007). Autoencoders have been used in various forms to address problems in machine translation (Chandar et al., 2014; Tu et al., 2017), speech processing (Elman & Zipser, 1987; Zeiler et al., 2013), and computer vision (Rifai et al., 2011; Larsson, 2017), to name just a few areas. Denoising autoencoders (DAEs) are an extension of autoencoders which learn latent features by reconstructing data from *corrupted* versions of the inputs (Vincent et al., 2008). Although this corruption step typically leads to improved performance over standard autoencoders, a theoretical understanding of its effects remains incomplete. In this paper, we provide new insights into the inner workings of DAEs by analysing the learning dynamics of linear DAEs.

We specifically build on the work of Saxe et al. (2013a;b), who studied the learning dynamics of deep linear networks in a supervised regression setting. By analysing the gradient descent weight update steps as time-dependent differential equations (in the limit as the learning rate approaches a small value), Saxe et al. (2013a) were able to derive exact solutions for the learning trajectory of these networks as a function of training time. Here we extend their approach to linear DAEs. To do this, we use the expected reconstruction loss over the noise distribution as an objective (requiring a different decomposition of the input covariance) as a tractable way to incorporate noise into our analytic solutions. This approach yields exact equations which can predict the learning trajectory of a linear DAE.

Our work here shares the motivation of many recent studies (Advani & Saxe, 2017; Pennington & Worah, 2017; Pennington & Bahri, 2017; Nguyen & Hein, 2017; Dinh et al., 2017; Louart et al., 2017; Swirszcz et al., 2017; Lin et al., 2017; Neyshabur et al., 2017; Soudry & Hoffer, 2017; Pennington et al., 2017) working towards a better theoretical understanding of neural networks and their behaviour. Although we focus here on a theory for *linear* networks, such networks have learning dynamics that are in fact *nonlinear*. Furthermore, analyses of linear networks have also proven useful in understanding the behaviour of nonlinear neural networks (Saxe et al., 2013a; Advani & Saxe, 2017).

First we introduce linear DAEs (§2). We then derive analytic expressions for their nonlinear learning dynamics (§3), and verify our solutions in simulations (§4) which show how noise can influence the shape of the loss surface and change the rate of convergence for gradient descent optimisation. We also find that an appropriate amount of noise can help DAEs ignore low variance directions in the input while learning the reconstruction mapping. In the remainder of

[1]Computer Science Division, Stellenbosch University, South Africa [2]CSIR/SU Centre for Artificial Intelligence Research, [3]Department of Electrical and Electronic Engineering, Stellenbosch University, South Africa. Correspondence to: Steve Kroon <kroon@sun.ac.za>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

[*]Code to reproduce all the results in this paper is available at: https://github.com/arnupretorius/lindaedynamics_icml2018

Stellenbosch University https://scholar.sun.ac.za

the paper, we compare DAEs to standard regularised autoencoders and show that our theoretical predictions match both simulations (§5) and experimental results on MNIST and CIFAR-10 (§6). We specifically find that while the noise in a DAE has an equivalent effect to standard weight decay, the DAE exhibits faster learning dynamics. We also show that our observations hold qualitatively for nonlinear DAEs.

2. Linear Denoising Autoencoders

We first give the background of linear DAEs. Given training data consisting of pairs $\{(\tilde{\mathbf{x}}_i, \mathbf{x}_i), i = 1, ..., N\}$, where $\tilde{\mathbf{x}}$ represents a corrupted version of the training data $\mathbf{x} \in \mathbb{R}^D$, the reconstruction loss for a single hidden layer DAE with activation function ϕ is given by

$$\mathcal{L} = \frac{1}{2N} \sum_{i=1}^{N} ||\mathbf{x}_i - W_2 \phi(W_1 \tilde{\mathbf{x}}_i)||^2.$$

Here, $W_1 \in \mathbb{R}^{H \times D}$ and $W_2 \in \mathbb{R}^{D \times H}$ are the weights of the network with hidden dimensionality H. The learned feature representations correspond to the latent variable $\mathbf{z} = \phi(W_1 \tilde{\mathbf{x}})$.

To corrupt an input $\mathbf{x}$, we sample a noise vector ϵ, where each component is drawn i.i.d. from a pre-specified noise distribution with mean zero and variance s^2. We define the corrupted version of the input as $\tilde{\mathbf{x}} = \mathbf{x} + \epsilon$. This ensures that the expectation over the noise remains unbiased, i.e. $\mathbb{E}_\epsilon(\tilde{\mathbf{x}}) = \mathbf{x}$.

Restricting our scope to linear neural networks, with $\phi(a) = a$, the loss in expectation over the noise distribution is

$$\mathbb{E}_\epsilon [\mathcal{L}] = \frac{1}{2N} \sum_{i=1}^{N} ||\mathbf{x}_i - W_2 W_1 \mathbf{x}_i||^2 + \frac{s^2}{2} \text{tr}(W_2 W_1 W_1^T W_2^T), \quad (1)$$

See the supplementary material for the full derivation.

3. Learning Dynamics of Linear DAEs

Here we derive the learning dynamics of linear DAEs, beginning with a brief outline to build some intuition.

The weight update equations for a linear DAE can be formulated as time-dependent differential equations in the limit as the gradient descent learning rate becomes small (Saxe et al., 2013a). The task of an ordinary (undercomplete) linear autoencoder is to learn the identity mapping that reconstructs the original input data. The matrix corresponding to this learned map will essentially be an approximation of the full identity matrix that is of rank equal to the input dimension. It turns out that tracking the temporal updates of this mapping represents a difficult problem that involves dealing with coupled differential equations, since both the on-diagonal and off-diagonal elements of the weight matrices need to be considered in the approximation dynamics at each time step.

To circumvent this issue and make the analysis tractable, we follow the methodology introduced in Saxe et al. (2013a), which is to: (1) decompose the input covariance matrix using an eigenvalue decomposition; (2) rotate the weight matrices to align with these computed directions of variation; and (3) use an orthogonal initialisation strategy to diagonalise the composite weight matrix $W = W_2 W_1$. The important difference in our setting, is that additional constraints are brought about through the injection of noise.

The remainder of this section outlines this derivation for the exact solutions to the learning dynamics of linear DAEs.

3.1. Gradient descent update

Consider a continuous time limit approach to studying the learning dynamics of linear DAEs. This is achieved by choosing a sufficiently small learning rate α for optimising the loss in (1) using gradient descent. The update for W_1 in a single gradient descent step then takes the form of a time-dependent differential equation

$$\begin{aligned} \tau \frac{d}{dt} W_1 &= \sum_{i=1}^{N} W_2^T \left(\mathbf{x}_i \mathbf{x}_i^T - W_2 W_1 \mathbf{x}_i \mathbf{x}_i^T \right) \\ &\quad - \varepsilon W_2^T W_2 W_1 \\ &= W_2^T \left(\Sigma_{xx} - W_2 W_1 \Sigma_{xx} \right) - \varepsilon W_2^T W_2 W_1. \end{aligned}$$

Here t is the time measured in epochs, $\tau = \frac{N}{\alpha}, \varepsilon = N s^2$ and $\Sigma_{xx} = \sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T$, represents the input covariance matrix. Let the eigenvalue decomposition of the input covariance be $\Sigma_{xx} = V \Lambda V^T$, where V is an orthogonal matrix and denote the eigenvalues $\lambda_j = [\Lambda]_{jj}$, with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D$. The update can then be rewritten as

$$\begin{aligned} \tau \frac{d}{dt} W_1 &= W_2^T V \left(\Lambda - V^T W_2 W_1 V \Lambda \right) V^T \\ &\quad - \varepsilon W_2^T W_2 W_1. \end{aligned}$$

The weight matrices can be rotated to align with the directions of variation in the input by performing the rotations $\overline{W}_1 = W_1 V$ and $\overline{W}_2 = V^T W_2$. Following a similar derivation for W_2, the weight updates become

$$\begin{aligned} \tau \frac{d}{dt} \overline{W}_1 &= \overline{W}_2^T \left(\Lambda - \overline{W}_2 \overline{W}_1 \Lambda \right) - \varepsilon \overline{W}_2^T \overline{W}_2 \overline{W}_1 \\ \tau \frac{d}{dt} \overline{W}_2 &= \left(\Lambda - \overline{W}_2 \overline{W}_1 \Lambda \right) \overline{W}_1^T - \varepsilon \overline{W}_2 \overline{W}_1 \overline{W}_1^T. \end{aligned}$$

3.2. Orthogonal initialisation and scalar dynamics

To decouple the dynamics, we can set $W_2 = V D_2 R^T$ and $W_1 = R D_1 V^T$, where R is an arbitrary orthogonal matrix

Stellenbosch University https://scholar.sun.ac.za

and D_2 and D_1 are diagonal matrices. This results in the product of the realigned weight matrices

$$\overline{W}_2\overline{W}_1 = V^T V D_2 R^T R D_1 V^T V = D_2 D_1$$

to become diagonal. The updates now reduce to the following scalar dynamics that apply independently to each pair of diagonal elements w_{1j} and w_{2j} of D_1 and D_2 respectively:

$$\tau \frac{d}{dt} w_{1j} = w_{2j}\lambda_j \left(1 - w_{2j}w_{1j}\right) - \varepsilon w_{2j}^2 w_{1j} \tag{2}$$

$$\tau \frac{d}{dt} w_{2j} = w_{1j}\lambda_j \left(1 - w_{2j}w_{1j}\right) - \varepsilon w_{2j} w_{1j}^2. \tag{3}$$

Note that the same dynamics stem from gradient descent on the loss given by

$$\ell = \sum_{j=1}^{D} \frac{\lambda_j}{2\tau}(1 - w_{2j}w_{1j})^2 + \sum_{j=1}^{D} \frac{\varepsilon}{2\tau}(w_{2j}w_{1j})^2. \tag{4}$$

By examining (4), it is evident that the degree to which the first term will be reduced will depend on the magnitude of the associated eigenvalue λ_j. However, for directions in the input covariance Σ_{xx} with relatively little variation the decrease in the loss from learning the identity map will be negligible and is likely to result in overfitting (since little to no signal is being captured by these eigenvalues). The second term in (4) is the result of the input corruption and acts as a suppressant on the magnitude of the weights in the learned mapping. Our interest is to better understand the interplay between these two terms during learning by studying their scalar learning dynamics.

3.3. Exact solutions to the dynamics of learning

As noted above, the dynamics of learning are dictated by the value of $w = w_2 w_1$ over time. An expression can be derived for $w(t)$ by using a hyperbolic change of coordinates in (2) and (3), letting θ parameterise points along a dynamics trajectory represented by the conserved quantity $w_2^2 - w_1^2 = \pm c_0$. This relies on the fact that ℓ is invariant under a scaling of the weights such that $w = (w_1/c)(cw_2) = w_2 w_1$ for any constant c (Saxe et al., 2013a). Starting at any initial point (w_1, w_2) the dynamics are

$$w(t) = \frac{c_0}{2}\sinh\left(\theta_t\right), \tag{5}$$

with

$$\theta_t = 2\tanh^{-1}\left[\frac{(1-E)\left(\zeta^2 - \beta^2 - 2\beta\delta\right) - 2(1+E)\zeta\delta}{(1-E)\left(2\beta + 4\delta\right) - 2(1+E)\zeta}\right]$$

where $\beta = c_0\left(1 + \frac{\varepsilon}{\lambda}\right)$, $\zeta = \sqrt{\beta^2 + 4}$, $\delta = \tanh\left(\frac{\theta_0}{2}\right)$ and $E = e^{\zeta\lambda t/\tau}$. Here θ_0 depends on the initial weights w_1 and w_2 through the relationship $\theta_0 = \sinh^{-1}(2w/c_0)$. The derivation for θ_t involves rewriting $\tau\frac{d}{dt}w$ in terms of θ, integrating over the interval θ_0 to θ_t, and finally rearranging terms to get an expression for $\theta(t) \equiv \theta_t$ (see the supplementary material for full details). To derive the learning dynamics for different noise distributions, the corresponding ε must be computed and used to determine β and ζ. For example, sampling noise from a Gaussian distribution such that $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$, gives $\varepsilon = N\sigma^2$. Alternatively, if ϵ is distributed according to a zero-mean Laplace distribution with scale parameter b, then $\varepsilon = 2Nb^2$.

4. The Effects of Noise: a Simulation Study

Since the expression for the learning dynamics of a linear DAE in (5) evolve independently for each direction of variation in the input, it is enough to study the effect that noise has on learning for a single eigenvalue λ. To do this we trained a scalar linear DAE to minimise the loss $\ell_\lambda = \frac{\lambda}{2}(1 - w_2 w_1)^2 + \frac{\varepsilon}{2}(w_2 w_1)^2$ with $\lambda = 1$ using gradient descent. Starting from several different randomly initialised weights w_1 and w_2, we compare the simulated dynamics with those predicted by equation (5). The top row in Figure 1 shows the exact fit between the predictions and numerical simulations for different noise levels, $\varepsilon = 0, 1, 5$.

The trajectories in the top row of Figure 1 converge to the optimal solution at different rates depending on the amount of injected noise. Specifically, adding more noise results in faster convergence. However, the trade-off in (4) ensures that the fixed point solution also diminishes in magnitude.

To gain further insight, we also visualise the associated loss surfaces for each experiment in the bottom row of Figure 1. Note that even though the scalar product $w_2 w_1$ defines a linear mapping, the minimisation of ℓ_λ with respect to w_1 and w_2 is a non-convex optimisation problem. The loss surfaces in Figure 1 each have an unstable saddle point at $w_2 = w_1 = 0$ (red star) with all remaining fixed points lying on a minimum loss manifold (cyan curve). This manifold corresponds to the different possible combinations of w_2 and w_1 that minimise ℓ_λ. The paths that gradient descent follow from various initial starting weights down to points situated on the manifold are represented by dashed orange lines.

For a fixed value of λ, adding noise warps the loss surface making steeper slopes and pulling the minimum loss manifold in towards the saddle point. Therefore, steeper descent directions cause learning to converge at a faster rate to fixed points that are smaller in magnitude. This is the result of a sharper curving loss surface and the minimum loss manifold lying closer to the origin.

We can compute the fixed point solution for any pair of initial starting weights (not on the saddle point) by taking

Stellenbosch University https://scholar.sun.ac.za

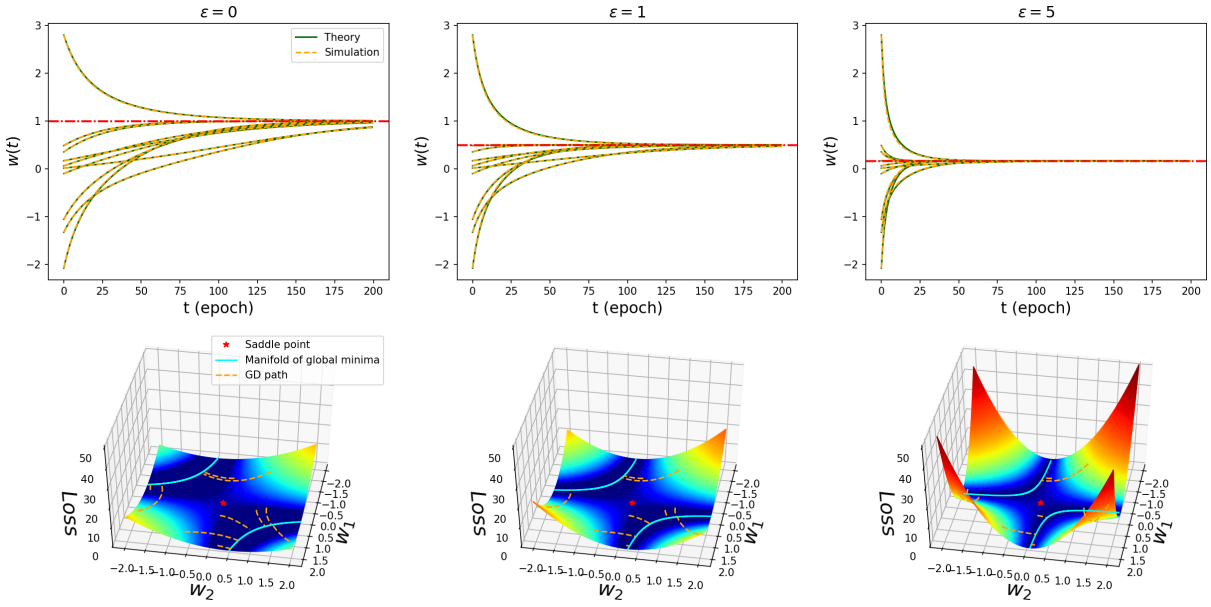


Figure 1. Learning dynamics, loss surface and gradient descent paths for linear denoising autoencoders. **Top**: Learning dynamics for each simulated run (dashed orange lines) together with the theoretically predicted learning dynamics (solid green lines). The red line in each plot indicates the final value of the resulting fixed point solution w^*. **Bottom**: The loss surface corresponding to the loss $\ell_\lambda = \frac{\lambda}{2}(1 - w_2 w_1)^2 + \frac{\varepsilon}{2}(w_2 w_1)^2$ for $\lambda = 1$, as well as the gradient descent paths (dashed orange lines) for randomly initialised weights. The cyan hyperbolas represent the global minimum loss manifold that corresponds to all possible combinations of w_2 and w_1 that minimise ℓ_λ. **Left**: $\varepsilon = 0, w^* = 1$. **Middle**: $\varepsilon = 1, w^* = 0.5$. **Right**: $\varepsilon = 5, w^* = 1/6$.

the derivative

$$\frac{d\ell_\lambda}{dw} = -\frac{\lambda}{\tau}(1 - w) + \frac{\varepsilon}{\tau}w,$$

and setting it equal to zero to find $w^* = \frac{\lambda}{\lambda+\varepsilon}$. This solution reveals the interaction between the input variance associated with λ and the noise ε. For large eigenvalues for which $\lambda \gg \varepsilon$, the fixed point will remain relatively unaffected by adding noise, i.e., $w^* \approx 1$. In contrast, if $\lambda \ll \varepsilon$, the noise will result in $w^* \approx 0$. This means that over a distribution of eigenvalues, an appropriate amount of noise can help a DAE to ignore low variance directions in the input data while learning the reconstruction. In a practical setting, this motivates the tuning of noise levels on a development set to prevent overfitting.

5. The Relationship Between Noise and Weight Decay

It is well known that adding noise to the inputs of a neural network is equivalent to a form of regularisation (Bishop, 1995). Therefore, to further understand the role of noise in linear DAEs we compare the dynamics of noise to those of explicit regularisation in the form of *weight decay* (Krogh & Hertz, 1992). The reconstruction loss for a linear weight decayed autoencoder (WDAE) is given by

$$\frac{1}{2N}\sum_{i=1}^{N} ||\mathbf{x}_i - W_2 W_1 \mathbf{x}_i||^2 + \frac{\gamma}{2}\left(||W_1||^2 + ||W_2||^2\right) \quad (6)$$

where γ is the penalty parameter that controls the amount of regularisation applied during learning. Provided that the weights of the network are initialised to be small, it is also possible (see supplementary material) to derive scalar dynamics of learning from (6) as

$$w_\gamma(t) = \frac{\xi E_\gamma}{E_\gamma - 1 + \xi/w_0}, \quad (7)$$

where $\xi = (1 - N\gamma/\lambda)$ and $E_\gamma = e^{2\xi t/\tau}$.

Figure 2 compares the learning trajectories of linear DAEs and WDAEs over time (as measured in training epochs) for $\lambda = 2.5, 1, 0.5$ and 0.1. The dynamics for both noise and weight decay exhibit a sigmoidal shape with an initial period of inactivity followed by rapid learning, finally reaching a plateau at the fixed point solution. Figure 2 illustrates that the learning time associated with an eigenvalue is negatively correlated with its magnitude. Thus, the eigenvalue corresponding to the largest amount of variation explained is the quickest to escape inactivity during learning.

The colour intensity of the lines in Figure 2 correspond to the amount of noise or regularisation applied in each run,

Stellenbosch University https://scholar.sun.ac.za

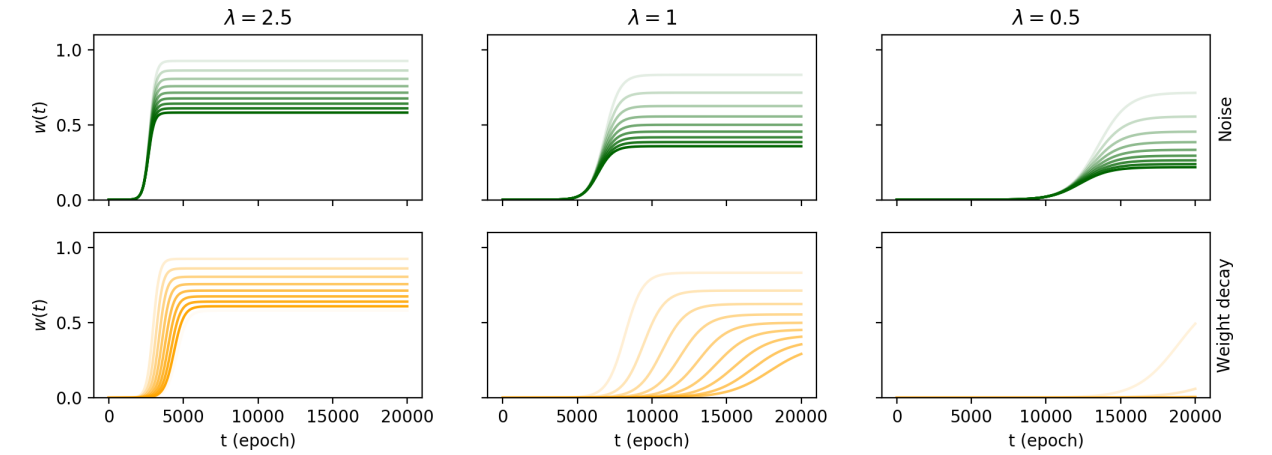


Figure 2. Theoretically predicted learning dynamics for noise compared to weight decay for linear autoencoders. **Top**: Noise dynamics (green), darker line colours correspond to larger amounts of added noise. **Bottom**: Weight decay dynamics (orange), darker line colours correspond to larger amounts of regularisation. **Left to right**: Eigenvalues $\lambda = 2.5, 1$ and 0.5 associated with high to low variance.

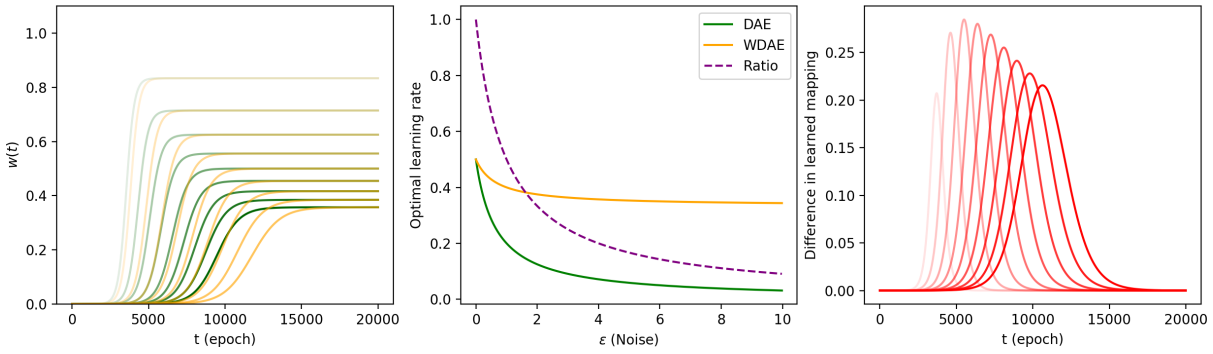


Figure 3. Learning dynamics for optimal discrete time learning rates ($\lambda = 1$). **Left**: Dynamics of DAEs (green) vs. WDAEs (orange), where darker line colours correspond to larger amounts noise or weigh decay. **Middle**: Optimal learning rate as a function of noise ε for DAEs, and for WDAEs using an equivalent amount of regularisation $\gamma = \lambda\varepsilon/(\lambda + \varepsilon)$. **Right**: Difference in mapping over time.

with darker lines indicating larger amounts. In the continuous time limit with equal learning rates, when compared with noise dynamics, weight decay experiences a delay in learning such that the initial inactive period becomes extended for every eigenvalue, whereas adding noise has no effect on learning time. In other words, starting from small weights, noise injected learning is capable of providing an equivalent regularisation mechanism to that of weight decay in terms of a constrained fixed point mapping, but with zero time delay.

However, this analysis does not take into account the practice of using well-tuned stable learning rates for discrete optimisation steps. We therefore consider the impact on training time when using optimised learning rates for each approach. By using second order information from the Hessian as in Saxe et al. (2013a), (here of the expected reconstruction loss with respect to the scalar weights), we relate the optimal learning rates for linear DAEs and WDAEs, where each optimal rate is inversely related to the amount of noise/regularisation applied during training (see supplementary material). The ratio of the optimal DAE rate to that for the WDAE is

$$R = \frac{2\lambda + \gamma}{2\lambda + 3\varepsilon}. \tag{8}$$

Note that the ratio in (8) will essentially be equal to one for eigenvalues that are significantly larger than both ε and γ, with deviations from unity only manifesting for smaller values of λ.

Furthermore, weight decay and noise injected learning result in equivalent scalar solutions when their parameters are related by $\gamma = \frac{\lambda\varepsilon}{\lambda+\varepsilon}$ (see supplementary material). This leads to the following two observations. First, it shows that adding noise during learning can be interpreted as a form of weight decay where the penalty parameter γ adapts to each direction of variation in the data. In other words, noise essentially makes use of the statistical structure of

Stellenbosch University https://scholar.sun.ac.za

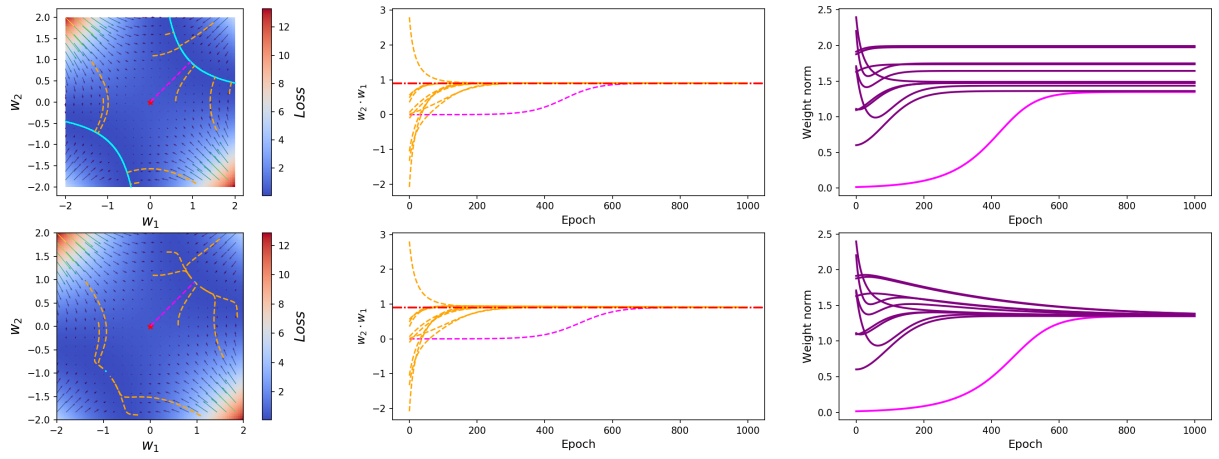


Figure 4. The effect of noise versus weight decay on the norm of the weights during learning. **Left**: Two-dimensional loss surface $\ell_\lambda = \frac{\lambda}{2}(1 - w_2 w_1)^2 + \frac{\varepsilon}{2}(w_2 w_1)^2 + \frac{\gamma}{2}(w_2^2 + w_1^2)$. Gradient descent paths (orange/magenta dashed lines), minimum loss manifold (cyan curves), saddle point (red star). **Middle**: Simulated learning dynamics. **Right**: Norm of the weights over time for each simulated run. **Top**: Noise with $\lambda = 1, \varepsilon = 0.1$ and $\gamma = 0$. **Bottom**: Weight decay with $\lambda = 1, \varepsilon = 0$ and $\gamma = \lambda(0.1)/(\lambda + 0.1) = 0.091$. The magenta line in each plot corresponds to a simulated run with small initialised weights.

the input data to influence the amount of shrinkage that is being applied in various directions during learning. Second, together with (8), we can theoretically compare the learning dynamics of DAEs and WDAEs, when both equivalent regularisation and the relative differences in optimal learning rates are taken into account.

The effects of optimal learning rates (for $\lambda = 1$), are shown in Figure 3. DAEs still exhibit faster dynamics (left panel), even when taking into account the difference in the learning rate as a function of noise, or equivalent weight decay (middle panel). In addition, for equivalent regularisation effects, the ratio of the optimal rates R can be shown to be a monotonically decreasing function of the noise level, where the rate of decay depends on the size of λ. This means that for any amount of added noise, the DAE will require a slower learning rate than that of the WDAE. Even so, a faster rate for the WDAE does not seem to compensate for its slower dynamics and the difference in learning time is also shown to grow as more noise (regularisation) is applied during training (right panel).

5.1. Exploiting invariance in the loss function

A primary motivation for weight decay as a regulariser is that it provides solutions with smaller weight norms, producing smoother models that have better generalisation performance. Figure 4 shows the effect of noise (top row) compared to weight decay (bottom row) on the norm of the weights during learning. Looking at the loss surface for weight decay (bottom left panel), the penalty on the size of the weights acts by shrinking the minimum loss manifold down from a long curving valley to a single point (associated with a small norm solution). Interestingly, this results in gradient descent following a trajectory towards an "invisible" minimum loss manifold similar to the one associated with noise. However, once on this manifold, weight decay begins to exploit invariances in the loss function to changes in the weights, so as to move along the manifold down towards smaller norm solutions. This means that even when the two approaches learn the exact same mapping over time (as shown by the learning dynamics in the middle column of Figure 4), additional epochs will cause weight decay to further reduce the size of the weights (bottom right panel). This happens in a stage-like manner where the optimisation first focuses on reducing the reconstruction loss by learning the optimal mapping and then reduces the regularisation loss through invariance.

5.2. Small weight initialisation and early stopping

It is common practice to initialise the weights of a network with small values. In fact, this strategy has recently been theoretically shown to help, along with early stopping, to ensure good generalisation performance for neural networks in certain high-dimensional settings (Advani & Saxe, 2017). In our analysis however, what we find interesting about small weight initialisation is that it removes some of the differences in the learning behaviour of DAEs compared to regularised autoencoders that use weight decay.

To see this, the magenta lines in Figure 4 show the learning dynamics for the two approaches where the weights of both the networks were initialised to small random starting values. The learning dynamics are almost identical in terms of their temporal trajectories and have equal fixed

Stellenbosch University https://scholar.sun.ac.za

points. However, what is interesting is the implicit regularisation that is brought about through the small initialisation. By starting small and making incremental updates to the weights, the scalar solution in both cases end up being equal to the minimum norm solution. In other words, the path that gradient descent takes from the initialisation to the minimum loss manifold, reaches the manifold where the norm of the weights happen to also be small. This means that the second phase of weight decay (where the invariance of the loss function would be exploited to reduce the regularisation penalty), is not only no longer necessary, but also does not result in a norm that is appreciably smaller than that obtained by learning with added noise. Therefore in this case, learning with explicit regularisation provides no additional benefit over that of learning with noise in terms of reducing the norm of the weights during training.

When initialising small, early stopping can also serve as a form of implicit regularisation by ensuring that the weights do not change past the point where the validation loss starts to increase (Bengio et al., 2007). In the context of learning dynamics, early stopping for DAEs can be viewed as a method that effectively selects only the directions of variation deemed useful for generalisation during reconstruction, considering the remaining eigenvalues to carry no additional signal.

6. Experimental Results

To verify the dynamics of learning on real-world data sets we compared theoretical predictions with actual learning on MNIST and CIFAR-10. In our experiments we considered the following linear autoencoder networks: a regular AE, a WDAE and a DAE.

For MNIST, we trained each autoencoder with small randomly initialised weights, using $N = 50000$ training samples for 5000 epochs, with a learning rate $\alpha = 0.01$ and a hidden layer width of $H = 256$. For the WDAE, the penalty parameter was set at $\gamma = 0.5$ and for the DAE, $\sigma^2 = 0.5$. The results are shown in Figure 5 (left column).

The theoretical predictions (solid lines) in Figure 5 show good agreement with the actual learning dynamics (points). As predicted, both regularisation (orange) and noise (green) suppress the fixed point value associated with the different eigenvalues and, whereas regularisation delays learning (fewer fixed points are reached by the WDAE during training when compared to the DAE), the use of noise has no effect on training time.

Similar agreement is shown for CIFAR-10 in the right column of Figure 5. Here, we trained each network with small randomly initialised weights using $N = 30000$ training samples for 5000 epochs, with a learning rate $\alpha = 0.001$ and a hidden dimension $H = 512$. For the WDAE, the

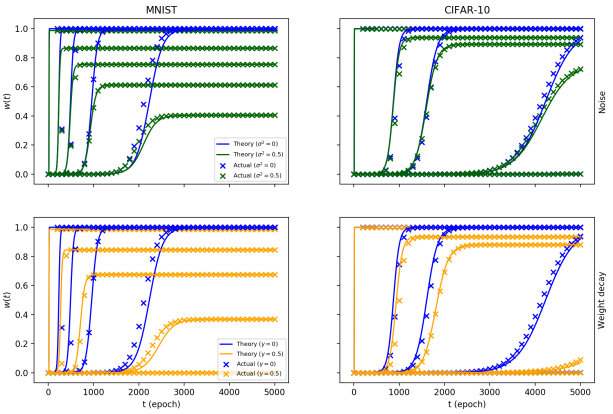


Figure 5. Learning dynamics for MNIST and CIFAR-10. Solid lines represent theoretical dynamics and 'x' markers simulated dynamics. Shown are the mappings associated with the set of eigenvalues $\{\lambda_i, i = 1, 4, 8, 16, 32\}$, where the remaining eigenvalues were excluded to improve readability. **Top**: Noise: AE (blue) vs. DAE with $\sigma^2 = 0.5$ (green). **Bottom**: Weight decay: AE (blue) vs. WDAE with $\gamma = 0.5$ (orange). **Left**: MNIST. **Right**: CIFAR-10.

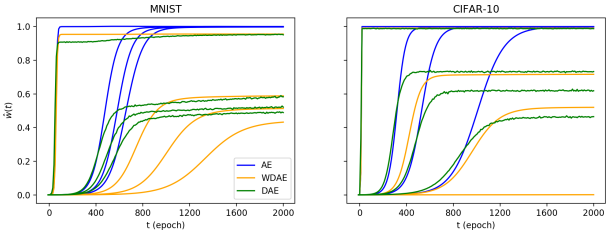


Figure 6. Learning dynamics for nonlinear networks using ReLU activation. AE (blue), WDAE (orange) and DAE (green). Shown are the mappings associated with the first four eigenvalues, i.e. $\{\lambda_i, i = 1, 2, 3, 4\}$. **Left**: MNIST **Right**: CIFAR-10.

penalty parameter was set at $\gamma = 0.5$ and for the DAE, $\sigma^2 = 0.5$.

Next, we investigated whether these dynamics are at least also qualitatively present in nonlinear autoencoder networks. Figure 6 shows the dynamics of learning for nonlinear AEs, WDAEs and DAEs, using ReLU activations, trained on MNIST ($N = 50000$) and CIFAR-10 ($N = 30000$) with equal learning rates. For the DAE, the input was corrupted using sampled Gaussian noise with mean zero and $\sigma^2 = 3$. For the WDAE, the amount of weight decay was manually tuned to $\gamma = 0.0045$, to ensure that both autoencoders displayed roughly the same degree of regularisation in terms of the fixed points reached. During the course of training, the identity mapping associated with each eigenvalue was estimated (see supplementary material), at equally spaced intervals of size 10 epochs.

The learning dynamics are qualitatively similar to the dy-

Stellenbosch University https://scholar.sun.ac.za

namics observed in the linear case. Both noise and weight decay result in a shrinkage of the identity mapping associated with each eigenvalue. Furthermore, in terms of the number of training epochs, the DAE is seen to learn as quickly as a regular AE, whereas the WDAE incurs a delay in learning time. Although these experimental results stem from a single training run for each autoencoder, we note that wall-clock times for training may still differ because DAEs require some additional time for sampling noise. Similar results were observed when using a tanh nonlinearity and are provided in the supplementary material.

7. Related Work

There have been many studies aiming to provide a better theoretical understanding of DAEs. Vincent et al. (2008) analysed DAEs from several different perspectives, including manifold learning and information filtering, by establishing an equivalence between different criteria for learning and the original training criterion that seeks to minimise the reconstruction loss. Subsequently, Vincent (2011) showed that under a particular set of conditions, the training of DAEs can also be interpreted as a type of score matching. This connection provided a probabilistic basis for DAEs. Following this, a more in-depth analysis of DAEs as a possible generative model suitable for arbitrary loss functions and multiple types of data was given by Bengio et al. (2013).

In contrast to a probabilistic understanding of DAEs, we present here an analysis of the learning process. Specifically inspired by Saxe et al. (2013a), as well as by earlier work on supervised neural networks (Opper, 1988; Sanger, 1989; Baldi & Hornik, 1989; Saad & Solla, 1995), we provide a theoretical investigation of the temporal behaviour of linear DAEs using derived equations that exactly describe their dynamics of learning. Specifically for the linear case, the squared error loss for the reconstruction contractive autoencoder (RCAE) introduced in Alain & Bengio (2014) is equivalent to the expected loss (over the noise) for the DAE. Therefore, the learning dynamics described in this paper also apply to linear RCAEs.

For our analysis to be tractable we used a marginalised reconstruction loss where the gradient descent dynamics are viewed in expectation over the noise distribution. Whereas our motivation is analytical in nature, marginalising the reconstruction loss tends to be more commonly motivated from the point of view of learning useful and robust feature representations at a significantly lower computational cost (Chen et al., 2014; 2015). This approach has also been investigated in the context of supervised learning (van der Maaten et al., 2013; Wang & Manning, 2013; Wager et al., 2013). Also related to our work is the analysis by Poole et al. (2014), who showed that training autoencoders with noise (added at different levels of the network architecture), is closely connected to training with explicit regularisation and proposed a marginalised noise framework for noisy autoencoders.

8. Conclusion and Future Work

This paper analysed the learning dynamics of linear denoising autoencoders (DAEs) with the aim of providing a better understanding of the role of noise during training. By deriving exact time-dependent equations for learning, we showed how noise influences the shape of the loss surface as well as the rate of convergence to fixed point solutions. We also compared the learning behaviour of added input noise to that of weight decay, an explicit form of regularisation. We found that while the two have similar regularisation effects, the use of noise for regularisation results in faster training. We compared our theoretical predictions with actual learning dynamics on real-world data sets, observing good agreement. In addition, we also provided evidence (on both MNIST and CIFAR-10) that our predictions hold qualitatively for nonlinear DAEs.

This work provides a solid basis for further investigation. Our analysis could be extended to nonlinear DAEs, potentially using the recent work on nonlinear random matrix theory for neural networks (Pennington & Worah, 2017; Louart et al., 2017). Our findings indicate that appropriate noise levels help DAEs ignore low variance directions in the input; we also obtained new insights into the training time of DAEs. Therefore, future work might consider how these insights could actually be used for tuning noise levels and predicting the training time of DAEs. This would require further validation and empirical experiments, also on other datasets. Finally, our analysis only considers the training dynamics, while a better understanding of generalisation and what influences the quality of feature representations during testing, are also of prime importance.

Acknowledgements

We would like to thank Andrew Saxe for early discussions that got us interested in this work, as well as the reviewers for insightful comments and suggestions. We would like to thank the CSIR/SU Centre for Artificial Intelligence Research (CAIR), South Africa, for financial support. AP would also like to thank the MIH Media Lab at Stellenbosch University and Praelexis (Pty) Ltd for providing stimulating working environments for a portion of this work.

Stellenbosch University https://scholar.sun.ac.za

References

Advani, M. S. and Saxe, A. M. High-dimensional dynamics of generalization error in neural networks. *arXiv:1710.03667*, 2017.

Alain, G. and Bengio, Y. What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, 15(1):3563–3593, 2014.

Baldi, P. and Hornik, K. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58, 1989.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*, pp. 153–160, 2007.

Bengio, Y., Yao, L., Alain, G., and Vincent, P. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, pp. 899–907, 2013.

Bishop, C. M. Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, 7(1):108–116, 1995.

Chandar, S., Lauly, S., Larochelle, H., Khapra, M., Ravindran, B., Raykar, V. C., and Saha, A. An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems*, pp. 1853–1861, 2014.

Chen, M., Weinberger, K., Sha, F., and Bengio, Y. Marginalized denoising auto-encoders for nonlinear representations. In *International Conference on Machine Learning*, pp. 1476–1484, 2014.

Chen, M., Weinberger, K., Xu, Z., and Sha, F. Marginalizing stacked linear denoising autoencoders. *Journal of Machine Learning Research*, 16:3849–3875, 2015.

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. Sharp minima can generalize for deep nets. *arXiv:1703.04933*, 2017.

Elman, J. L. and Zipser, D. Learning the hidden structure of speech. *Journal of the Acoustic Society of America*, 83: 1615–1626, 1987.

Krogh, A. and Hertz, J. A. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems*, pp. 950–957, 1992.

Larsson, G. Discovery of visual semantics by unsupervised and self-supervised representation learning. *arXiv:1708.05812*, 2017.

Lin, H. W., Tegmark, M., and Rolnick, D. Why does deep and cheap learning work so well? *Journal of Statistical Physics*, 168:1223–1247, 2017.

Louart, C., Liao, Z., and Couillet, R. A random matrix approach to neural networks, 2017. arXiv:1702.05419v2.

Neyshabur, B., Tomioka, R., Salakhutdinov, R., and Srebro, N. Geometry of optimization and implicit regularization in deep learning. *arXiv:1705.03071*, 2017.

Nguyen, Q. and Hein, M. The loss surface of deep and wide neural networks. *arXiv:1704.08045*, 2017.

Opper, M. Learning times of neural networks: exact solution for a perceptron algorithm. *Physical Review A*, 38(7): 3824, 1988.

Pennington, J. and Bahri, Y. Geometry of neural network loss surfaces via random matrix theory. In *International Conference on Machine Learning*, pp. 2798–2806, 2017.

Pennington, J. and Worah, P. Nonlinear random matrix theory for deep learning. In *Advances in Neural Information Processing Systems*, pp. 2634–2643, 2017.

Pennington, J., Schoenholz, S., and Ganguli, S. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In *Advances in Neural Information Processing Systems*, pp. 4788–4798, 2017.

Poole, B., Sohl-Dickstein, J., and Ganguli, S. Analyzing noise in autoencoders and deep networks. *arXiv:1406.1831*, 2014.

Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. Contractive auto-encoders: Explicit invariance during feature extraction. In *International Conference on Machine Learning*, pp. 833–840, 2011.

Saad, D. and Solla, S. A. Exact solution for on-line learning in multilayer neural networks. *Physical Review Letters*, 74(21):4337, 1995.

Sanger, T. D. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2(6):459–473, 1989.

Saxe, A. M., McClelland, J. L., and Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv:1312.6120*, 2013a.

Saxe, A. M., McClelland, J. L., and Ganguli, S. Learning hierarchical category structure in deep neural networks. In *Proceedings of the Cognitive Science Society*, pp. 1271–1276, 2013b.

Soudry, D. and Hoffer, E. Exponentially vanishing sub-optimal local minima in multilayer neural networks. *arXiv:1702.05777*, 2017.

Stellenbosch University https://scholar.sun.ac.za

Swirszcz, G., Czarnecki, W. M., and Pascanu, R. Local minima in training of neural networks. *arXiv:1611.06310*, 2017.

Tu, Z., Liu, Y., Shang, L., Liu, X., and Li, H. Neural machine translation with reconstruction. In *AAAI Conference on Artificial Intelligence*, pp. 3097–3103, 2017.

van der Maaten, L., Chen, M., Tyree, S., and Weinberger, K. Learning with marginalized corrupted features. In *International Conference on Machine Learning*, pp. 410–418, 2013.

Vincent, P. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning*, pp. 1096–1103, 2008.

Wager, S., Wang, S., and Liang, P. S. Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems*, pp. 351–359, 2013.

Wang, S. and Manning, C. Fast dropout training. In *International Conference on Machine Learning*, pp. 118–126, 2013.

Zeiler, M., Ranzato, M., Monga, R., Mao, M., Yang, K., Le, Q., Nguyen, P., Senior, A., Vanhoucke, V., Dean, J., and Hinton, G. E. On rectified linear units for speech processing. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.

Stellenbosch University https://scholar.sun.ac.za

Supplementary material

The following section provides detail omitted in the paper regarding the derivation of certain equations as well as additional comments.

A. Expected loss for linear DAEs

We derive the expected reconstruction loss over the noise distribution as presented in (1) in the paper. The expected loss can be written as

$$\mathbb{E}_\epsilon[\mathcal{L}] = \frac{1}{2N}\sum_{i=1}^{N}\mathbb{E}_\epsilon\left[||\mathbf{x}_i - W_2W_1\tilde{\mathbf{x}}_i||^2\right].$$

where $\tilde{\mathbf{x}}_i = \mathbf{x}_i + \epsilon_i$, with ϵ sampled from an isotropic noise distribution with component variance s^2. Let $SE(\tilde{\mathbf{x}}_i) = ||\mathbf{x}_i - W_2W_1\tilde{\mathbf{x}}_i||^2$ and $M = W_2W_1$. Then

$$\begin{aligned}\mathbb{E}_\epsilon\left[SE(\tilde{\mathbf{x}}_i)\right] &= \mathbb{E}_\epsilon\left[||(I-M)\mathbf{x}_i + M(\mathbf{x}_i - \tilde{\mathbf{x}}_i)||^2\right] \\ &= SE(\mathbf{x}_i) + \mathbb{E}_\epsilon\left[||M(\mathbf{x}_i - \tilde{\mathbf{x}}_i)||^2\right]\end{aligned}$$

because the cross product terms vanish, since $\mathbb{E}_\epsilon[\tilde{\mathbf{x}}_i] = \mathbf{x}_i$:

$$\begin{aligned}0 &= \mathbb{E}_\epsilon\left[\mathbf{x}_i^T(I-M)^TM(\mathbf{x}_i - \tilde{\mathbf{x}}_i)\right] \\ &= \mathbb{E}_\epsilon\left[(\mathbf{x}_i - \tilde{\mathbf{x}}_i)^TM^T(I-M)\mathbf{x}_i\right].\end{aligned}$$

We also have that

$$\begin{aligned}||M(\mathbf{x}_i - \tilde{\mathbf{x}}_i)||^2 &= (\mathbf{x}_i - \tilde{\mathbf{x}}_i)^TM^TM(\mathbf{x}_i - \tilde{\mathbf{x}}_i) \\ &= \text{tr}\left[(\mathbf{x}_i - \tilde{\mathbf{x}}_i)^TM^TM(\mathbf{x}_i - \tilde{\mathbf{x}}_i)\right] \\ &= \text{tr}\left[M(\mathbf{x}_i - \tilde{\mathbf{x}}_i)(\mathbf{x}_i - \tilde{\mathbf{x}}_i)^TM^T\right] \\ &= \text{tr}\left[M\epsilon_i\epsilon_i^TM^T\right]\end{aligned}$$

due to the invariance of the trace under cycle permutation of products. Therefore, in expectation over the noise we have

$$\mathbb{E}_\epsilon\left[||M(\mathbf{x}_i - \tilde{\mathbf{x}}_i)||^2\right] = \text{tr}\left[M(s^2I)M^T\right],$$

and as a result

$$\begin{aligned}\mathbb{E}_\epsilon\left[\mathcal{L}\right] = \frac{1}{2N}\sum_{i=1}^{N}&||\mathbf{x}_i - W_2W_1\mathbf{x}_i||^2 \\ &+ \frac{s^2}{2}\text{tr}\left(W_2W_1W_1^TW_2^T\right).\end{aligned}$$

B. Learning dynamics for linear DAEs

We derive the expression for the learning dynamics of a linear DAE as presented in (5) in the paper. As departure point, we start by examining the expected scalar update equations over the noise model for a small learning rate α, which can be written as

$$\begin{aligned}\tau\frac{d}{dt}w_1 &= w_2(\lambda - w_2w_1\lambda) - \varepsilon w_2^2w_1 \\ \tau\frac{d}{dt}w_2 &= w_1(\lambda - w_2w_1\lambda) - \varepsilon w_2w_1^2.\end{aligned}$$

where $\tau = \frac{N}{\alpha}$, with N representing the number of training samples. Define $w = w_2w_1$ and using the product rule the update for w then becomes

$$\begin{aligned}\tau\frac{d}{dt}w &= \tau[w_1\frac{d}{dt}w_2 + w_2\frac{d}{dt}w_1] \\ &= w_1^2(\lambda - w_2w_1(\lambda+\varepsilon)) + w_2^2(\lambda - w_2w_1(\lambda+\varepsilon)) \\ &= (\lambda - w(\lambda+\varepsilon))(w_1^2 + w_2^2). \qquad (9)\end{aligned}$$

Next we make the following hyperbolic change of coordinates

$$w_1 = \sqrt{c_0}\text{sinh}\left(\frac{\theta}{2}\right), w_2 = \sqrt{c_0}\text{cosh}\left(\frac{\theta}{2}\right), \text{ for } w_1^2 < w_2^2$$

$$w_1 = \sqrt{c_0}\text{cosh}\left(\frac{\theta}{2}\right), w_2 = \sqrt{c_0}\text{sinh}\left(\frac{\theta}{2}\right), \text{ for } w_1^2 > w_2^2,$$

where θ parameterises points along the dynamics trajectory represented by $w_2^2 - w_1^2 = \pm c_0$ (Saxe et al., 2013a). Note that with this change of coordinates we obtain

$$\begin{aligned}w &= c_0\text{cosh}\left(\frac{\theta}{2}\right)\text{sinh}\left(\frac{\theta}{2}\right) \\ &= c_0\left(\frac{e^{\frac{\theta}{2}} + e^{-\frac{\theta}{2}}}{2}\right)\left(\frac{e^{\frac{\theta}{2}} - e^{-\frac{\theta}{2}}}{2}\right) \\ &= \frac{c_0}{2}\left(\frac{e^{\theta} - e^{-\theta}}{2}\right) \\ &= \frac{c_0}{2}\text{sinh}(\theta),\end{aligned}$$

so that

$$dw = \frac{c_0}{2}\text{cosh}(\theta)d\theta.$$

Similarly,

$$\begin{aligned}w_2^2 + w_1^2 &= c_0\text{cosh}^2\left(\frac{\theta}{2}\right) + c_0\text{sinh}^2\left(\frac{\theta}{2}\right) \\ &= c_0\left(\frac{e^{\frac{\theta}{2}} + e^{-\frac{\theta}{2}}}{2}\right)^2 + c_0\left(\frac{e^{\frac{\theta}{2}} - e^{-\frac{\theta}{2}}}{2}\right)^2 \\ &= \frac{c_0}{4}\left(e^{\theta} + 2 + e^{-\theta} + e^{\theta} - 2 + e^{-\theta}\right) \\ &= c_0\left(\frac{e^{\theta} + e^{-\theta}}{2}\right) \\ &= c_0\text{cosh}(\theta)\end{aligned}$$

Plugging these results into the update for w given in (9), yields

$$\frac{\tau c_0\text{cosh}(\theta)}{2}\frac{d\theta}{dt} = \left(\lambda - \frac{c_0}{2}\text{sinh}(\theta)(\lambda+\varepsilon)\right)c_0\text{cosh}(\theta),$$

and as a result,

$$\tau\frac{d\theta}{dt} = \lambda\left(2 - \beta\text{sinh}(\theta)\right),$$

Stellenbosch University https://scholar.sun.ac.za

where $\beta = c_0\left(1+\frac{\varepsilon}{\lambda}\right)$. To solve for t, we write

$$t = \int_{\theta_0}^{\theta_f} \frac{\tau}{\lambda\left(2-\beta\sinh(\theta)\right)} d\theta$$

and integrate:

$$t = \frac{\tau}{\zeta\lambda}\left[\ln\left(\frac{\zeta+\beta+2\tanh(\frac{\theta}{2})}{\zeta-\beta-2\tanh(\frac{\theta}{2})}\right)\right]_{\theta_0}^{\theta_f}$$

where $\zeta = \sqrt{\beta^2+4}$ and initial parameter value $\theta_0 = \sinh^{-1}(2w/c_0)$. Let $\delta_0 = \tanh\left(\frac{\theta_0}{2}\right)$ and $\delta_f = \tanh\left(\frac{\theta_f}{2}\right)$, then

$$t = \frac{\tau}{\lambda\zeta}\ln\frac{\left(\zeta+\beta+2\delta_f\right)\left(\zeta-\beta-2\delta_0\right)}{\left(\zeta-\beta-2\delta_f\right)\left(\zeta+\beta+2\delta_0\right)},$$

so that

$$e^{\lambda\zeta t/\tau} = \frac{\left(\zeta+\beta+2\delta_f\right)\left(\zeta-\beta-2\delta_0\right)}{\left(\zeta-\beta-2\delta_f\right)\left(\zeta+\beta+2\delta_0\right)}.$$

Multiplying by the denominator, expanding, and defining $E = e^{\lambda\zeta t/\tau}$, we obtain

$$\begin{aligned}
&-2E\delta_f\left(\zeta+\beta+2\delta_0\right)\\
&+E\left(\zeta^2+2\zeta\delta_0-\beta^2-2\beta\delta_0\right)\\
&=2\delta_f\left(\zeta-\beta-2\delta_0\right)\\
&+\left(\zeta^2-2\zeta\delta_0-\beta^2-2\beta\delta_0\right),
\end{aligned}$$

which yields

$$\begin{aligned}
&\delta_f\left((1-E)\left(2\beta+4\delta_0\right)-2(E+1)\zeta\right)\\
&=(1-E)\left(\zeta^2-\beta^2-2\beta\delta_0\right)-2(1+E)\zeta\delta_0.
\end{aligned}$$

Solving for $\theta_f(t)$, we obtain the hyperbolic parameter equation

$$\theta_f(t) = 2\tanh^{-1}\left[\frac{(1-E)\left(\zeta^2-\beta^2-2\beta\delta\right)-2(1+E)\zeta\delta}{(1-E)\left(2\beta+4\delta\right)-2(1+E)\zeta}\right]$$

where $\delta = \tanh\left(\frac{\theta_0}{2}\right)$. Using

$$w(t) = \frac{c_0}{2}\sinh\left(\theta_t\right),$$

(where $\theta_t = \theta_f(t)$) to track the weight trajectory gives equation (5) in the paper.

C. Learning dynamics for linear WDAEs

We derive the expression for the learning dynamics of a linear WDAE as presented in (7) in the paper. Reconstruction loss with weight decay gives the scalar loss associated with an eigenvalue λ as

$$\ell_\gamma = \frac{\lambda}{2\tau}(1-w_2w_1)^2 + \frac{N\gamma}{2\tau}(w_1^2+w_2^2),$$

where γ is the penalty parameter that controls the amount of regularisation that is being applied. The update equations for the weights then follow as

$$\begin{aligned}
\tau\frac{d}{dt}w_1 &= w_2(\lambda - w_2w_1\lambda) - N\gamma w_1\\
\tau\frac{d}{dt}w_2 &= w_1(\lambda - w_2w_1\lambda) - N\gamma w_2,
\end{aligned}$$

assuming the initial $w_2 = w_1$ (which holds approximately for small initial values), we have for $w = w_2w_1$ that

$$\begin{aligned}
\tau\frac{d}{dt}w &= 2w(\lambda - w\lambda) - 2N\gamma w\\
&= 2w(\lambda - N\gamma - w\lambda).
\end{aligned}$$

Thus,

$$\begin{aligned}
t &= \int_{w_0}^{w_f}\frac{\tau}{2w(\lambda-N\gamma-w\lambda)}dw\\
&= \frac{\tau}{2}\left[\frac{\ln(w)-\ln(\lambda-N\gamma-w\lambda)}{\lambda-N\gamma}\right]_{w_0}^{w_f}\\
&= \frac{\tau}{2(\lambda-N\gamma)}\ln\left(\frac{w_f(\lambda-N\gamma-w_0\lambda)}{w_0(\lambda-N\gamma-w_f\lambda)}\right).
\end{aligned}$$

Then solving for w_f gives

$$w_f(t) = \frac{\xi E_\gamma}{E_\gamma - 1 + \xi/w_0},$$

where $E_\gamma = e^{2\xi t/\tau}$ and $\xi = (1-N\gamma/\lambda)$.

D. Optimal learning rates

We derive expressions for the optimal learning rates for linear DAEs and WDAEs as presented in (8) in the paper. First, consider the expected scalar DAE loss

$$\ell_\varepsilon = \frac{\lambda}{2\tau}(1-w_2w_1)^2 + \frac{\varepsilon}{2\tau}(w_2w_1)^2.$$

The Hessian of ℓ_ε is given by

$$H = \begin{bmatrix}\frac{\partial^2\ell_\varepsilon}{\partial w_1^2} & \frac{\partial^2\ell_\varepsilon}{\partial w_1w_2}\\ \frac{\partial^2\ell_\varepsilon}{\partial w_2w_1} & \frac{\partial^2\ell_\varepsilon}{\partial w_2^2}\end{bmatrix},$$

where

$$\begin{aligned}
\frac{\partial^2\ell_\varepsilon}{\partial w_1^2} &= \frac{w_2^2}{\tau}(\lambda+\varepsilon),\\
\frac{\partial^2\ell_\varepsilon}{\partial w_2^2} &= \frac{w_1^2}{\tau}(\lambda+\varepsilon),\\
\frac{\partial^2\ell_\varepsilon}{\partial w_1w_2} &= \frac{\partial^2\ell_\varepsilon}{\partial w_2w_1} = \frac{2w_2w_1}{\tau}(\lambda+\varepsilon) - \frac{\lambda}{\tau}.
\end{aligned}$$

Stellenbosch University https://scholar.sun.ac.za

Now, if we assume $w_2 = w_1$, and let $a = \frac{\partial^2 \ell_\varepsilon}{\partial w_1^2} = \frac{\partial^2 \ell_\varepsilon}{\partial w_2^2}$ and $b = \frac{\partial^2 \ell_\varepsilon}{\partial w_2 w_1}$, the eigenvalues for the Hessian can be shown to be $\lambda_H = a - b$ or $\lambda_H = a + b$. The second order update for a single weight w at time t is then given by

$$w^{t+1} = w^t - \left(\frac{\partial \ell_\varepsilon}{\partial w^t}\right) / \lambda_H,$$

where the maximum λ_H, is when $w_2 = w_1 = 1$, such that

$$\begin{aligned}\lambda_H &= \frac{1}{\tau}(\lambda + \varepsilon) + \frac{2}{\tau}(\lambda + \varepsilon) - \frac{\lambda}{\tau} \\ &= \frac{2\lambda + 3\varepsilon}{\tau}.\end{aligned}$$

Therefore, the optimal learning rate is

$$\alpha_\varepsilon = 1/\lambda_H = \frac{\tau}{2\lambda + 3\varepsilon}.$$

For WDAEs with penalty parameter γ, a very similar derivation gives

$$\alpha_\gamma = \frac{\tau}{2\lambda + \gamma}.$$

Taking the ratio of the optimal DAE rate to that for the WDAE gives

$$R = \frac{\alpha_\varepsilon}{\alpha_\gamma} = \frac{2\lambda + \gamma}{2\lambda + 3\varepsilon}.$$

E. Equivalent scalar solutions

In Section 4 of the paper, the DAE fixed point solution is shown to be

$$w_\varepsilon^* = \frac{\lambda}{\lambda + \varepsilon}.$$

Now if $w = w_2 w_1$ and $w_2 = w_1$, then for WDAE we have that the scalar loss is given by

$$\ell_\gamma = \frac{\lambda}{2\tau}(1 - w)^2 + \frac{\gamma}{\tau} w,$$

and

$$\frac{\partial \ell_\gamma}{\partial w} = -\frac{\lambda}{\tau}(1 - w) + \frac{\gamma}{\tau}.$$

Setting the above equal to zero and solving gives

$$w_\gamma^* = 1 - \gamma/\lambda.$$

To obtain the value of γ for which the two fixed points are equal, we set $w_\gamma^* = w_\varepsilon^*$ and solve for γ to find

$$\gamma = \frac{\lambda\varepsilon}{\lambda + \varepsilon}.$$

F. Estimated dynamics for nonlinear networks

The dynamics for the nonlinear networks trained in Figure 6 in the paper were estimated using the following approach. First, compute

$$\Sigma_{xx} = \sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T = V \Lambda V^T,$$

using an eigen-decomposition giving eigenvalues $\lambda_j, j = 1, ..., D$. Then at regular intervals compute

$$\hat{\Sigma}_{xx}(t) = \sum_{i=1}^{N} \mathbf{x}_i \hat{\mathbf{x}}_i(t)^T,$$

where $\hat{\mathbf{x}}(t)$ is the estimated reconstruction of input at time t generated by the autoencoder network. Finally, using the following rotation to obtain the diagonal matrix

$$\hat{\Lambda}(t) = V^T \hat{\Sigma}_{xx}(t) V,$$

where the diagonal contains the estimated eigenvalues $\hat{\lambda}_j(t)$, we can compute an estimate for the identity mapping associated with each eigenvalue as $\hat{\lambda}_j(t)/\lambda_j \in [0, 1]$.

G. Learning dynamics for tanh autoencoder networks

We investigated the dynamics of learning for nonlinear AEs, WDAEs and DAEs, using tanh activations.

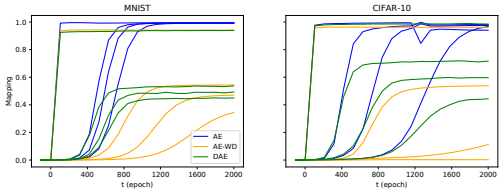


Figure 7. Learning dynamics for nonlinear networks using tanh activation. AE (blue), WDAE (orange) and DAE (green). **Left**: MNIST **Right**: CIFAR-10.

Figure 7 shows the dynamics for these networks trained on MNIST ($N = 50000$) and CIFAR-10 ($N = 30000$) with equal learning rates. For the DAE, the input was corrupted using sampled Gaussian noise with mean zero and $\sigma^2 = 2$. For the WDAE, the amount of weight decay was set to $\gamma = 0.0045$. During the course of training, the identity mapping associated with each eigenvalue was estimated using the approach described in Section F, at equally spaced intervals of size 100 epochs.

Stellenbosch University https://scholar.sun.ac.za

2.4 Discussion

In **this chapter**, we shed some light on how noise regularisation affects the learning dynamics of DAEs. Injecting noise into an AE has empirically been shown to produce more useful latent feature representations. However, we show that noise regularisation can also help AEs learn the input statistics in time that is proportional to the variance along the different directions in the input space. This results in higher variance directions (important for reconstruction) being learned first, whereas lower variance directions (which are less relevant) incurring significant delays in learning time. Moreover, this change in the learning dynamics could have advantages in terms of training speed over other regularisation methods such as weight decay.

We comment on a key assumption underlying our interpretation of the findings. It is often a reasonable assumption that higher variance directions correspond to signal rather than noise, but this is of course not always the case. Directions that monotonically decrease in variance do not necessarily correspond to directions that are monotonically less important. Therefore, even though we observe learning times ordered according to the amount of variance explained, in practice, intermediate directions with moderate amounts of variance might still be crucial for generalisation.

In our empirical experiments we did not elaborate on the specifics of weight initialisation. Therefore, we provide more information here. We initialised the network weights as follows: $w_{ij}^l \sim \mathcal{N}(0, \sigma_w^2)$ where w_{ij}^l represent the scalar weights in the weight matrix W_l for $l = 1, 2$. The scale of the weights σ_w^2 was set to 10^{-8}, which we considered to be small enough to approximate "equal" starting weights for the dynamics. To compare the theory with actual training, we track the dynamics of $w(t)$ in Equation (5) in the paper and compare it to the change in the eigenvalues of $W = W_2 W_1$ during training. We have to choose scalar starting values for w_1 and w_2 in Equation (5), i.e. choose $w(0)$. For this, we used the following procedure. We computed the eigenvalues of W at initialisation, which we considered to be approximations to the initial scalar products $w_2 w_1$ associated with each specific direction of variation —this would be exact if W were diagonal. However, an issue that arises is that W changes over time and we are unable to identify a priori which eigenvalue in the initial W will be associated with which direction of variation in the input covariance. Because of this, for each eigenvalue being tracked theoretically, we decided to set $w(0)$ as the mean of the eigenvalues of the initial W.

As demonstrated in the paper (e.g. Figures 1 and 4), weight initialisation can strongly influence the learning dynamics of a regularised autoencoder neural network. Moreover, weight initialisation interacts with regularisation in interesting ways, such as making the effects of noise more similar to that of weight decay.

In the **next chapter**, we specifically focus on initialisation for noise regularised *deep* non-linear neural networks. This will take us beyond the shallow linear case as well as into the more traditional supervised deep learning setting. We will investigate how noise interacts with initialisation as well as network depth and discover how to best initialise a neural network given that its inputs at various layers are corrupted by a pre-specified multiplicative noise distribution.

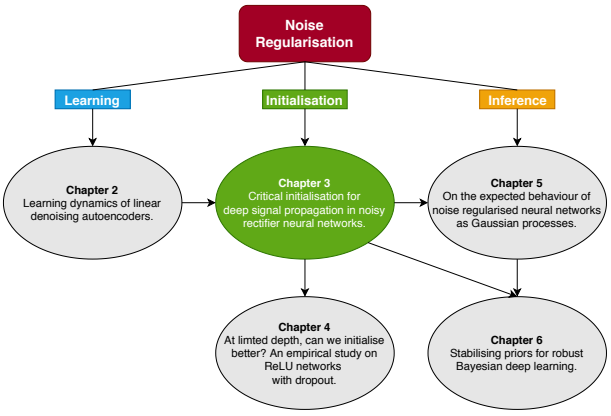
Stellenbosch University https://scholar.sun.ac.za

Chapter 3

Initialisation strategies for noise regularised neural networks

Paper
Pretorius, A., van Biljon, E., Kroon, S. & Kamper, H. Critical initialisation for deep signal propagation in noisy rectifier neural networks. *Advances in Neural Information Processing Systems (NeurIPS, 2018).*
Available at: http://papers.nips.cc/paper/7814-critical-initialisation-for-deep-signal-propagation-in-noisy-rectifier-neural-networks.pdf

Code
Pretorius, A., van Biljon, E., Kamper, H. (2018). Code for: Critical initialisation for deep signal propagation in noisy rectifier neural networks.
Available at: https://github.com/arnupretorius/noisysignalprop_neurips2018.



3.1 Introduction

The behaviour of a deep neural network during optimisation depends on its starting parameter values, referred to as its *initialisation*. Initialisation can be seen as a regulariser in its own right since it biases the network to certain regions of the hypothesis space. This bias can sometimes be the difference between successful training and a network not being able to train at all.

For a *deep* neural network, standard initialisation strategies are likely to be adversely affected by noise corruption. This is because if the scale of the weights is not adjusted to

Stellenbosch University https://scholar.sun.ac.za

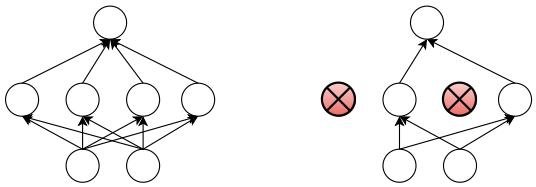


Figure 3.1: *Dropout.* **Left**: Network with no noise. **Right**: Network with dropout. The red crossed-out units represent deleted units.

compensate for the injected noise, the signal in a deep network can completely vanish at the start of training. Therefore, an appropriate initialisation strategy for noise regularised networks should factor in the amount of noise being injected into the network.

In this chapter, we consider general noise regularisation in deep neural networks, of which dropout (Srivastava *et al.*, 2014) is a special case. We derive new optimal initialisation strategies for noise regularised fully-connected feedforward neural networks that use rectifier activations, such as the popular rectified linear unit (ReLU). We also find that noise regularisation limits the depth to which ReLU networks are able to train, and that this depth limit depends on the amount of noise being injected into the network. We verify these findings using experiments on simulated as well as real-world data sets.

3.2 Background

Historically, deep neural networks were largely untrainable until the development of *greedy layer-wise unsupervised pretraining* (Bengio *et al.*, 2007). This technique used the autoencoder models investigated in Chapter 2 to learn useful representations for each layer in turn (using the learned representation from the previous layer as the input to the next), before training the entire network end-to-end. Pretraining can be viewed as an expensive form of network initialisation that reduces variance during optimisation (Erhan *et al.*, 2010). Stable variance initialisation has since motivated the development of simpler initialisation strategies that make training deep neural networks possible without any pretraining (Glorot and Bengio, 2010; He *et al.*, 2015). These initialisations have become standard practice in deep learning.

Standard initialisation strategies work by specifying an appropriate scale for the weights of the network. If the weights are too large, signal from the input will compound through the layers and explode or saturate in activation. If the weights are too small, the signal will vanish. The optimal scale for the weights is chosen exactly at the boundary between these two regimes of signal propagation.

It is important to note that the initialisation of a deep neural network may interact with other design aspects of the network. For instance, other regularisation mechanisms such as noise regularisation, may behave very differently depending on how the network is initialised.

Consider *dropout* (Srivastava *et al.*, 2014), an extremely successful form of noise regularisation for deep neural networks. Dropout regularises a network by randomly deleting hidden units during training (as shown in Figure 3.1), in an attempt to discourage learned co-dependencies between different units. Deleting hidden units is a form of noise injection, where the noise is sampled from a Bernoulli distribution. What makes dropout an effective strategy in deep neural networks is that it injects noise at *different levels of abstraction*. For example, consider the task of recognising a car in an image. Now, suppose

Stellenbosch University https://scholar.sun.ac.za

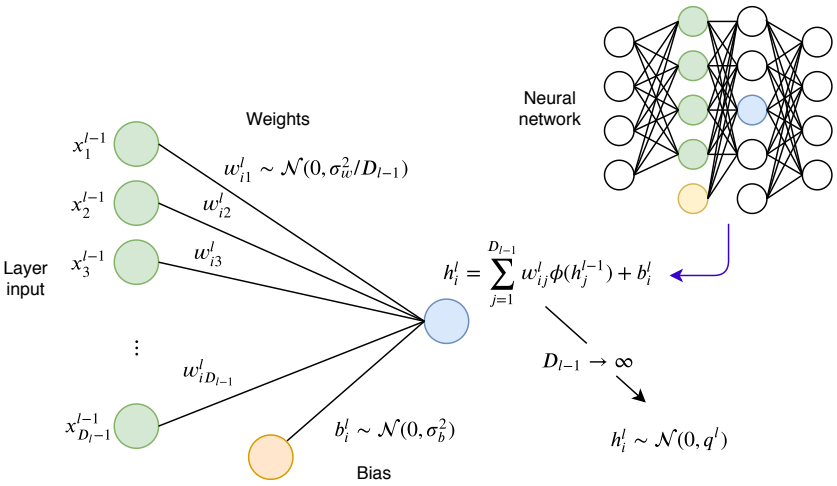


Figure 3.2: *Mean field theory*: unit pre-activations become Gaussian distributed in the large width limit.

a higher-level unit has learned to activate when a wheel is somewhere in the image, so as to indicate the presence of a car. If this unit gets corrupted with noise, the network will be forced to find other features useful for identifying a car, such as the headlights or doors. In this way, dropout encourages the network to make better use of the available input information.

3.2.1 Mean field theory for neural networks

Our analysis in this chapter, and in subsequent chapters, makes use of *mean field theory* for deep neural networks (Sompolinsky *et al.*, 1988; Poole *et al.*, 2016). Although this is discussed in more detail later, we briefly outline the basic idea here for convenience.

Consider a single scalar pre-activation h_i^l at a layer l of size D_l, with $i \in \{1, ..., D_l\}$.[1] This pre-activation is computed as a linear combination of the scalar activations $x_j^{l-1} = \phi(h_j^{l-1})$ in the previous layer, for $j = 1, ..., D_{l-1}$: we have

$$h_i^l = \sum_{j=1}^{D_{l-1}} w_{ij}^l \phi(h_j^{l-1}) + b_i^l, \tag{3.2.1}$$

where $\phi(\cdot)$ is the chosen activation function. This is illustrated in Figure 3.2. We assume that the initial weights and biases are sampled as follows, $w_{ij}^l \sim \mathcal{N}(0, \sigma_w^2/D_{l-1})$ and $b_i^l \sim \mathcal{N}(0, \sigma_b^2)$. Under these distributional assumptions for the parameters, we can compute the moments of h_i^l as

$$\mathbb{E}_{w,b}[h_i^l] = 0, \tag{3.2.2}$$

$$\mathbb{E}_{w,b}[(h_i^l)^2] = \sigma_w^2 \frac{1}{D_{l-1}} \sum_{j=1}^{D_{l-1}} \phi(h_j^{l-1})^2 + \sigma_b^2. \tag{3.2.3}$$

[1] We highlight a slight variation of notation. For this introduction, we use lower case letters (e.g. h_i) to indicate scalar values instead of a bold letter vector with a subscript to indicate indexing (selecting) a scalar value from this vector (e.g. $\mathbf{h}_i$), as is done in the paper, i.e. we use h_i instead of $\mathbf{h}_i$ for the same scalar value. Since the argument presented in the introduction is just in terms of scalar values, we felt this use of notation would be easier to follow.

Stellenbosch University https://scholar.sun.ac.za

The approach in mean field theory is to approximate the distribution of h_i^l by a Gaussian distribution, matching the above first and second moments. Specifically, we replace h_i^l with $\sqrt{q^l}z$, where $z \sim \mathcal{N}(0,1)$ and $q^l = \mathbb{E}_{w,b}[(h_i^l)^2]$. The argument for this substitution rests on the *central limit theorem*, where it is assumed that the sum in (3.2.1) tends towards a Gaussian distribution as the number of units in the previous layer becomes large, i.e. as $D_{l-1} \to \infty$. We can now use this argument recursively as it applies to the different units in each layer in turn to write

$$q^l = \sigma_w^2 \mathbb{E}_z \left[\phi(\sqrt{q^{l-1}}z)^2 \right] + \sigma_b^2, \tag{3.2.4}$$

where we replace the empirical average in (3.2.3) by an expectation over z. We use (3.2.4) to analyse the effect of initialisation on the variance dynamics of signal propagation through the layers of a network for a single input. A similar construction also allows us to analyse the dynamics of correlations between signals for different inputs through the layers, as well as the behaviour of error signal propagation during the backward pass.

3.3 Contribution statement

The idea to analyse the effects of noise at initialisation in nonlinear deep neural networks was my own. My decision to use mean field theory can be traced back to the work that inspired the previous paper, i.e. Saxe *et al.* (2014) (specifically, see Section 4 and Appendix G in the paper). This work was later extended by Poole *et al.* (2016) and Schoenholz *et al.* (2017) and largely influenced the analysis presented in this paper. In terms of contributions, I derived the theoretical results in the paper. Elan van Biljon and I ran all the experiments. We adapted large portions of the code that was made publicly available by Poole *et al.* (2016) for reproducing their experiments. I wrote the paper. Dr Steve Kroon provided technical feedback. Dr Herman Kamper gave general feedback as well as editorial suggestions that improved the final presentation of the paper.

Stellenbosch University https://scholar.sun.ac.za

Critical initialisation for deep signal propagation in noisy rectifier neural networks

Arnu Pretorius[*]
Computer Science Division
CAIR[†]
Stellenbosch University

Elan Van Biljon
Computer Science Division
Stellenbosch University

Steve Kroon
Computer Science Division
Stellenbosch University

Herman Kamper
Department of Electrical and Electronic Engineering
Stellenbosch University

Abstract

Stochastic regularisation is an important weapon in the arsenal of a deep learning practitioner. However, despite recent theoretical advances, our understanding of how noise influences signal propagation in deep neural networks remains limited. By extending recent work based on mean field theory, we develop a new framework for signal propagation in stochastic regularised neural networks. Our *noisy signal propagation* theory can incorporate several common noise distributions, including additive and multiplicative Gaussian noise as well as dropout. We use this framework to investigate initialisation strategies for noisy ReLU networks. We show that no critical initialisation strategy exists using additive noise, with signal propagation exploding regardless of the selected noise distribution. For multiplicative noise (e.g. dropout), we identify alternative critical initialisation strategies that depend on the second moment of the noise distribution. Simulations and experiments on real-world data confirm that our proposed initialisation is able to stably propagate signals in deep networks, while using an initialisation disregarding noise fails to do so. Furthermore, we analyse correlation dynamics between inputs. Stronger noise regularisation is shown to reduce the depth to which discriminatory information about the inputs to a noisy ReLU network is able to propagate, even when initialised at criticality. We support our theoretical predictions for these trainable depths with simulations, as well as with experiments on MNIST and CIFAR-10.[‡]

1 Introduction

Over the last few years, advances in network design strategies have made it easier to train large networks and have helped to reduce overfitting. These advances include improved weight initialisation strategies (Glorot and Bengio, 2010; Saxe et al., 2014; Sussillo and Abbott, 2014; He et al., 2015; Mishkin and Matas, 2016), non-saturating activation functions (Glorot et al., 2011) and stochastic regularisation techniques (Srivastava et al., 2014). Authors have noted, for instance, the critical dependence of successful training on noise-based methods such as dropout (Krizhevsky et al., 2012; Dahl et al., 2013).

[*]Correspondence: arnupretorius@gmail.com
[†]CSIR/SU Centre for Artificial Intelligence Research.
[‡]Code to reproduce all the results is available at https://github.com/ElanVB/noisy_signal_prop

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

Stellenbosch University https://scholar.sun.ac.za

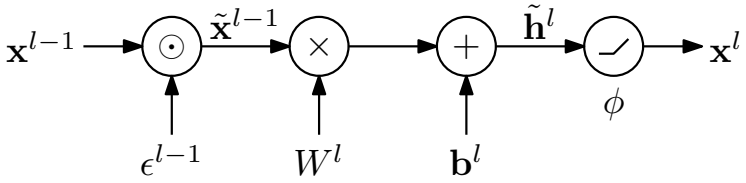


Figure 1: *Noisy layer recursion.* The input $\mathbf{x}^{l-1}$ from the previous layer gets corrupted by the sampled noise ϵ^{l-1}, either by vector addition or component-wise multiplication, producing the noisy inputs $\tilde{\mathbf{x}}^{l-1}$. The l^{th} layer's corrupted pre-activations are then computed by multiplication with the layer weight matrix W^l, followed by a vector addition of the biases $\mathbf{b}^l$. Finally, the inputs to the next layer are simply the activations of the current layer, *i.e.* $\mathbf{x}^l = \phi(\tilde{\mathbf{h}}^l)$.

In many cases, successful results arise only from effective combination of these advances. Despite this interdependence, our theoretical understanding of how these mechanisms and their interactions affect neural networks remains impoverished.

One approach to studying these effects is through the lens of deep neural signal propagation. By modelling the empirical input variance dynamics at the point of random initialisation, Saxe et al. (2014) were able to derive equations capable of describing how signal propagates in nonlinear fully connected feed-forward neural networks. This "mean field" theory was subsequently extended by Poole et al. (2016) and Schoenholz et al. (2017), in particular, to analyse signal correlation dynamics. These analyses highlighted the existence of a critical boundary at initialisation, referred to as the "edge of chaos". This boundary defines a transition between ordered (vanishing), and chaotic (exploding) regimes for neural signal propagation. Subsequently, the mean field approximation to random neural networks has been employed to analyse other popular neural architectures (Yang and Schoenholz, 2017; Xiao et al., 2018; Chen et al., 2018).

This paper focuses on the effect of noise on signal propagation in deep neural networks. Firstly we ask: How is signal propagation in deep neural networks affected by noise? To gain some insight into this question, we extend the mean field theory developed by Schoenholz et al. (2017) for the special case of dropout noise, into a generalised framework capable of describing the signal propagation behaviour of stochastically regularised neural networks for different noise distributions.

Secondly we ask: How much are current weight initialisation strategies affected by noise-induced regularisation in terms of their ability to initialise at a critical point for stable signal propagation? Using our derived theory, we investigate this question specifically for rectified linear unit (ReLU) networks. In particular, we show that no such critical initialisation exists for arbitrary zero-mean additive noise distributions. However, for multiplicative noise, such an initialisation is shown to be possible, given that it takes into account the amount of noise being injected into the network. Using these insights, we derive novel critical initialisation strategies for several different multiplicative noise distributions.

Finally, we ask: Given that a network is initialised at criticality, in what way does noise influence the network's ability to propagate useful information about its inputs? By analysing the correlation between inputs as a function of depth in random deep ReLU networks, we highlight the following: even though the statistics of individual inputs are able to propagate arbitrarily deep at criticality, *discriminatory information* about the inputs becomes lost at shallower depths as the noise in the network is increased. This is because in the later layers of a random noisy network, the internal representations from different inputs become uniformly correlated. Therefore, the application of noise regularisation directly limits the trainable depth of critically initialised ReLU networks.

2 Noisy signal propagation

We begin by presenting mean field equations for stochastically regularised fully connected feed-forward neural networks, allowing us to study noisy signal propagation for a variety of noise distributions. To understand how noise influences signal propagation in a random network given an input $\mathbf{x}^0 \in \mathbb{R}^{D_0}$, we inject noise into the model

$$\tilde{\mathbf{h}}^l = W^l(\mathbf{x}^{l-1} \odot \epsilon^{l-1}) + \mathbf{b}^l, \qquad \text{for } l = 1, ..., L \tag{1}$$

Stellenbosch University https://scholar.sun.ac.za

using the operator $\odot$ to denote either addition or multiplication where ϵ^l is an input noise vector, sampled from a pre-specified noise distribution. For additive noise, the distribution is assumed to be zero mean, for multiplicative noise distributions, the mean is assumed to be equal to one. The weights $W^l \in \mathbb{R}^{D_l \times D_{l-1}}$ and biases $\mathbf{b}^l \in \mathbb{R}^{D_l}$ are sampled i.i.d. from zero mean Gaussian distributions with variances σ_w^2/D_{l-1} and σ_b^2, respectively, where D_l denotes the dimensionality of the l^{th} hidden layer in the network. The hidden layer activations $\mathbf{x}^l = \phi(\tilde{\mathbf{h}}^l)$ are computed element-wise using an activation function $\phi(\cdot)$, for layers $l = 1, ..., L$. Figure 1 illustrates this recursive sequence of operations.

To describe forward signal propagation for the model in (1), we make use of the mean field approximation as in Poole et al. (2016) and analyse the statistics of the internal representations of the network in expectation over the parameters and the noise. Since the weights and biases are sampled from zero mean Gaussian distributions with pre-specified variances, we can approximate the distribution of the pre-activations at layer l, in the large width limit, by a zero mean Gaussian with variance

$$\tilde{q}^l = \sigma_w^2 \left\{ \mathbb{E}_z \left[\phi\left(\sqrt{\tilde{q}^{l-1}} z \right)^2 \right] \odot \mu_2^{l-1} \right\} + \sigma_b^2, \tag{2}$$

where $z \sim \mathcal{N}(0,1)$ (see Section A.1 in the supplementary material). Here, $\mu_2^l = \mathbb{E}_\epsilon[(\epsilon^l)^2]$ is the second moment of the noise distribution being sampled from at layer l. The initial input variance is given by $q^0 = \frac{1}{D_0}\mathbf{x}^0 \cdot \mathbf{x}^0$. Furthermore, to study the behaviour of a pair of signals from two different inputs, $\mathbf{x}^{0,a}$ and $\mathbf{x}^{0,b}$, passing through the network, we can compute the covariance at each layer as

$$\tilde{q}_{ab}^l = \sigma_w^2 \mathbb{E}_{z_1} \left[\mathbb{E}_{z_2} \left[\phi(\tilde{u}_1)\phi(\tilde{u}_2) \right] \right] + \sigma_b^2 \tag{3}$$

where $\tilde{u}_1 = \sqrt{\tilde{q}_{aa}^{l-1}} z_1$ and $\tilde{u}_2 = \sqrt{\tilde{q}_{bb}^{l-1}} \left[\tilde{c}^{l-1} z_1 + \sqrt{1 - (\tilde{c}^{l-1})^2} z_2 \right]$, with the correlation between inputs at layer l given by $\tilde{c}^l = \tilde{q}_{ab}^l / \sqrt{\tilde{q}_{aa}^l \tilde{q}_{bb}^l}$. Here, q_{aa}^l is the variance of $\tilde{\mathbf{h}}_j^{l,a}$ (see Section A.2 in the supplementary material for more details).

For the backward pass, we use the equations derived in Schoenholz et al. (2017) to describe error signal propagation.[1] In the context of mean field theory, the expected magnitude of the gradient at each layer can be shown to be proportional to the variance of the error, $\tilde{\delta}_i^l = \phi'(\tilde{\mathbf{h}}_i^l) \sum_{j=1}^{D_{l+1}} \tilde{\delta}_j^{l+1} W_{ji}^{l+1}$. This allows for the distribution of the error signal at layer l to be approximated by a zero mean Gaussian with variance

$$\tilde{q}_\delta^l = \tilde{q}_\delta^{l+1} \frac{D_{l+1}}{D_l} \sigma_w^2 \mathbb{E}_z \left[\phi'\left(\sqrt{\tilde{q}^l} z \right)^2 \right]. \tag{4}$$

Similarly, for noise regularised networks, the covariance between error signals can be shown to be

$$\tilde{q}_{ab,\delta}^l = \tilde{q}_{ab,\delta}^{l+1} \frac{D_{l+1}}{D_{l+2}} \sigma_w^2 \mathbb{E}_{z_1} \left[\mathbb{E}_{z_2} \left[\phi'(\tilde{u}_1)\phi'(\tilde{u}_2) \right] \right], \tag{5}$$

where $\tilde{u}_1$ and $\tilde{u}_2$ are defined as was done in the forward pass.

Equations (2)-(5) fully capture the relevant statistics that govern signal propagation for a random network during both the forward and the backward pass. In the remainder of this paper, we consider, as was done by Schoenholz et al. (2017), the following necessary condition for training: "for a random network to be trained information about the inputs should be able to propagate forward through the network, and information about the gradients should be able to propagate backwards through the network." The behaviour of the network at this stage depends on the choice of activation, noise regulariser and initial parameters. In the following section, we will focus on networks that use the Rectified Linear Unit (ReLU) as activation function. The chosen noise regulariser is considered a design choice left to the practitioner. Therefore, whether a random noisy ReLU network satisfies the above stated necessary condition for training largely depends on the starting parameter values of the network, *i.e.* its initialisation.

[1] It is, however, important to note that the derivation relies on the assumption that the weights used in the forward pass are sampled independently from those used during backpropagation.

Stellenbosch University https://scholar.sun.ac.za

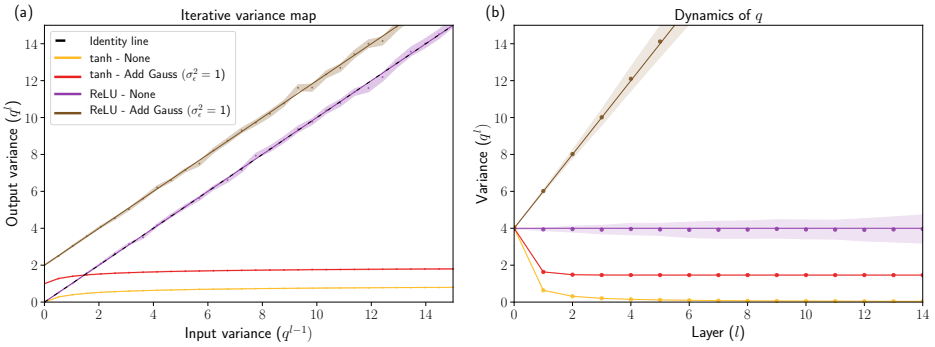


Figure 2: *Deep signal propagation with and without noise*. **(a)**: Iterative variance map. **(b)**: Variance dynamics during forward signal propagation. In (a) and (b), lines correspond to theoretical predictions and points to numerical simulations (means over 50 runs with shaded one standard deviation bounds), for noiseless tanh (yellow) and noiseless ReLU (purple) networks, as well as for noisy tanh (red) and noisy ReLU (brown) networks regularised using additive noise from a standard Gaussian. Both tanh networks use $(\sigma_w, \sigma_b) = (1, 0)$, the "Xavier" initialisation (Glorot and Bengio, 2010), while the ReLU networks use $(\sigma_w, \sigma_b) = (\sqrt{2}, 0)$ the "He" initialisation (He et al., 2015). In our experiments, we use network layers consisting of 1000 hidden units (see Section C in the supplementary material for more details on all our simulated experiments).

3 Critical initialisation for noisy rectifier networks

Unlike the tanh nonlinearity investigated in previous work (Poole et al., 2016; Schoenholz et al., 2017), rectifying activation functions such as ReLU are unbounded. This means that the statistics of signal propagation through the network is not guaranteed to naturally stabilise through saturating activations, as shown in Figure 2.

A point on the identity line in Figure 2 (a) represents a fixed point to the recursive variance map in equation (2). At a fixed point, signal will stably propagate through the remaining layers of the network. For tanh networks, such a fixed point always exists irrespective of the initialisation, or the amount of noise injected into the network. For ReLU networks, this is not the case. Consider the "He" initialisation (He et al., 2015) for ReLU, commonly used in practice. In (b), we plot the variance dynamics for this initialisation in purple and observe stable behaviour. But what happens when we inject noise into each network? In the case of tanh (shown in red), the added noise simply shifts the fixed point to a new stable value. However, for ReLU, the noise entirely destroys the fixed point for the "He" initialisation, making signal propagation unstable. This can be seen in (a), where the variance map for noisy ReLU (shown in brown) moves off the identity line entirely, causing the signal in (b) to explode.

Therefore, to investigate whether signal can stably propagate through a random *noisy* ReLU network, we examine (2) more closely, which for ReLU becomes (see Section B.1 in supplementary material)

$$\tilde{q}^l = \sigma_w^2 \left[\frac{\tilde{q}^{l-1}}{2} \odot \mu_2 \right] + \sigma_b^2. \tag{6}$$

For ease of exposition we assume equal noise levels at each layer, *i.e.* $\mu_2^l = \mu_2, \forall l$. A critical initialisation for a noisy ReLU network occurs when the tuple $(\sigma_w, \sigma_b, \mu_2)$ provides a fixed point $\tilde{q}^*$, to the recurrence in (6). This at least ensures that the statistics of individual inputs to the network will be preserved throughout the first forward pass. The existence of such a solution depends on the type of noise that is injected into the network. In the case of additive noise, $\tilde{q}^* = \sigma_w^2 \frac{1}{2}\tilde{q}^* + \mu_2 \sigma_w^2 + \sigma_b^2$, implying that the only critical point initialisation for non-zero $\tilde{q}^*$ is given by $(\sigma_w, \sigma_b, \mu_2) = (\sqrt{2}, 0, 0)$. Therefore, critical initialisation is not possible using any amount of zero-mean additive noise, regardless of the noise distribution. For multiplicative noise, $\tilde{q}^* = \sigma_w^2 \frac{1}{2}\tilde{q}^* \mu_2 + \sigma_b^2$, so the solution $(\sigma_w, \sigma_b, \mu_2) = \left(\sqrt{\frac{2}{\mu_2}}, 0, \mu_2\right)$ provides a critical initialisation for noise distributions with mean one and a non-zero second moment μ_2. For example, in the case of multiplicative Gaussian noise, $\mu_2 = \sigma_\epsilon^2 + 1$, yielding critical initialisation with $(\sigma_w, \sigma_b) = \left(\sqrt{\frac{2}{\sigma^2+1}}, 0\right)$. For dropout noise,

Stellenbosch University https://scholar.sun.ac.za

Table 1: Critical point initialisation for noisy ReLU networks.

DISTRIBUTION	P(ϵ)	μ_2	CRITICAL INITIALISATION
— ADDITIVE NOISE —			
GAUSSIAN	$\mathcal{N}(0, \sigma_\epsilon^2)$	σ_ϵ^2	$(\sigma_w, \sigma_b, \sigma_\epsilon) = (\sqrt{2}, 0, 0)$
LAPLACE	$Lap(0, \beta)$	$2\beta^2$	$(\sigma_w, \sigma_b, \beta) = (\sqrt{2}, 0, 0)$
— MULTIPLICATIVE NOISE —			
GAUSSIAN	$\mathcal{N}(1, \sigma_\epsilon^2)$	$(\sigma_\epsilon^2 + 1)$	$(\sigma_w, \sigma_b, \sigma_\epsilon) = \left(\sqrt{\frac{2}{\sigma_\epsilon^2+1}}, 0, \sigma_\epsilon\right)$
LAPLACE	$Lap(1, \beta)$	$(2\beta^2 + 1)$	$(\sigma_w, \sigma_b, \beta) = \left(\sqrt{\frac{2}{2\beta^2+1}}, 0, \beta\right)$
POISSON	$Poi(1)$	2	$(\sigma_w, \sigma_b, \lambda) = (1, 0, 1)$
DROPOUT	$\text{P}(\epsilon = \frac{1}{p}) = p,$ $\text{P}(\epsilon = 0) = 1 - p$	$\frac{1}{p}$	$(\sigma_w, \sigma_b, p) = (\sqrt{2p}, 0, p)$

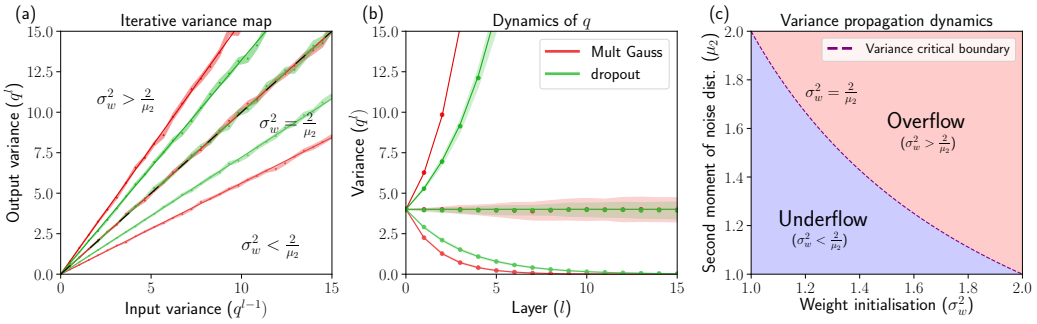


Figure 3: *Critical initialisation for noisy ReLU networks.* **(a)**: Iterative variance map. **(b)**: Variance dynamics during forward signal propagation. In (a) and (b), lines correspond to theoretical predictions and points to numerical simulations. Dropout ($p = 0.6$) is shown in green for different initialisations, $\sigma_w^2 = 2(0.6) = \frac{2}{\mu_2}$ (critical), $\sigma_w^2 = (1.15)^2 \frac{2}{(0.6)^{-1}} > \frac{2}{\mu_2}$ (exploding signal) and $\sigma_w^2 = (0.85)^2 \frac{2}{(0.6)^{-1}} < \frac{2}{\mu_2}$ (vanishing signal). Similarly, multiplicative Gaussian noise ($\sigma_\epsilon = 0.25$) is shown in red with $\sigma_w^2 = \frac{2}{(0.25)^2+1} = \frac{2}{\mu_2}$ (critical), $\sigma_w^2 = (1.25)^2 \frac{2}{\mu_2}$ (exploding) and $\sigma_w^2 = (0.75)^2 \frac{2}{\mu_2}$ (vanishing). **(c)**: Variance critical boundary for initialisation, separating numerical overflow and underflow signal propagation regimes.

$\mu_2 = 1/p$ (with p the probability of retaining a neuron); thus, to initialise at criticality, we must set $(\sigma_w, \sigma_b) = (\sqrt{2p}, 0)$. Table 1 summarises critical initialisations for some commonly used noise distributions. We also note that similar results can be derived for other rectifying activation functions; for example, for multiplicative noise the critical initialisation for parametric ReLU (PReLU) activations (with slope parameter α) is given by $(\sigma_w, \sigma_b, \mu_2) = \left(\sqrt{\frac{2}{\mu_2(\alpha^2+1)}}, 0, \mu_2\right)$.

To see the effect of initialising on or off the critical point for ReLU networks, Figure 3 compares the predicted versus simulated variance dynamics for different initialisation schemes. For schemes not initialising at criticality, the variance map in (a) no longer lies on the identity line and as a result the forward propagating signal in (b) either explodes, or vanishes. In contrast, the initialisations derived above lie on the critical boundary between these two extremes, as shown in (c) as a function of the noise. By compensating for the amount of injected noise, the signal corresponding to the initialisation $\sigma_w^2 = \frac{2}{\mu_2}$ is preserved in (b) throughout the entire forward pass, with roughly constant variance dynamics.

Stellenbosch University https://scholar.sun.ac.za

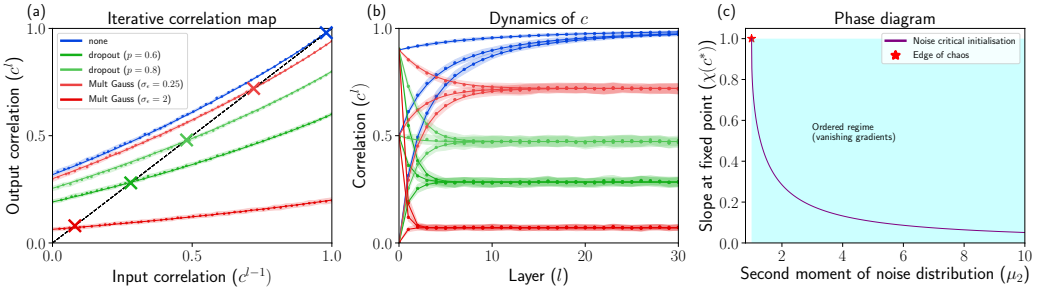


Figure 4: *Propagating correlation information in noisy ReLU networks.* **(a)**: Iterative correlation map with fixed points indicated by "X" marks on the identity line. **(b)**: Correlation dynamics during forward signal propagation. In (a) and (b), lines correspond to theoretical predictions and points to numerical simulations. All simulated networks were initialised at criticality for each noise type and level. **(c)**: Slope at the fixed point correlation as a function of the amount of noise injected into the network.

Next, we investigate the correlation dynamics between inputs. Assuming that (6) is at its fixed point $\tilde{q}^*$, which exists only if $\sigma_w^2 = \frac{2}{\mu_2}$, the correlation map for a noisy ReLU network is given by (see Section B.2 in supplementary material)

$$\tilde{c}^l = \frac{1}{\mu_2}\left\{\frac{\tilde{c}^{l-1}\sin^{-1}\left(\tilde{c}^{l-1}\right) + \sqrt{1-(\tilde{c}^{l-1})^2}}{\pi} + \frac{\tilde{c}^{l-1}}{2}\right\}. \tag{7}$$

Figure 4 plots this theoretical correlation map against simulated dynamics for different noise types and levels. For no noise, the fixed point c^* in (a) is situated at one (marked with an "X" on the blue line). The slope of the blue line indicates a non-decreasing function of the input correlations. After a certain depth, inputs end up perfectly correlated irrespective of their starting correlation, as shown in (b). In other words, random deep ReLU networks lose discriminatory information about their inputs as the depth of the network increases, even when initialised at criticality. When noise is added to the network, inputs decorrelate and c^* moves away from one. However, more importantly, correlation information in the inputs become lost at shallower depths as the noise level increases, as can be seen in (b).

How quickly a random network loses information about its inputs depends on the rate of convergence to the fixed point c^*. Using this observation, Schoenholz et al. (2017) derived so-called depth scales ξ_c, by assuming $|c^l - c^*| \sim e^{-l/\xi_c}$. These scales essentially control the feasible depth at which networks can be considered trainable, since they may still allow useful correlation information to propagate through the network. In our case, the depth scale for a noisy ReLU network under this assumption can be shown to be (see Section B.3 in supplementary material)

$$\xi_c = -1/\ln\left[\chi(c^*)\right], \tag{8}$$

where

$$\chi(c^*) = \frac{1}{\mu_2\pi}\left[\sin^{-1}(c^*) + \frac{\pi}{2}\right]. \tag{9}$$

The exponential rate assumption underlying the derivation of (8) is supported in Figure 5, where for different noise types and levels, we plot $|c^l - c^*|$ as a function of depth on a log-scale, with corresponding linear fits (see panels (a) and (c)). We then compare the theoretical depth scales from (8) to actual depth scales obtained through simulation (panels (b) and (d)), as a function of noise and observe a good fit for non-zero noise levels.[4] We thus find that noise limits the depth at which critically initialised ReLU networks are expected to perform well through training.

[4]We note Hayou et al. (2018) recently showed that the rate of convergence for noiseless ReLU networks is not exponential, but polynomial instead. Interestingly, keeping with the exponential rate assumption, we indeed find that the discrepancy between our theoretical depth scales from (8) and our simulated depth scales, is largest at very low noise levels. However, at more typical noise levels, such as a dropout rate of $p = 0.5$ for example, the assumption seems to provide a close fit, with good agreement between theory and simulation.

Stellenbosch University https://scholar.sun.ac.za

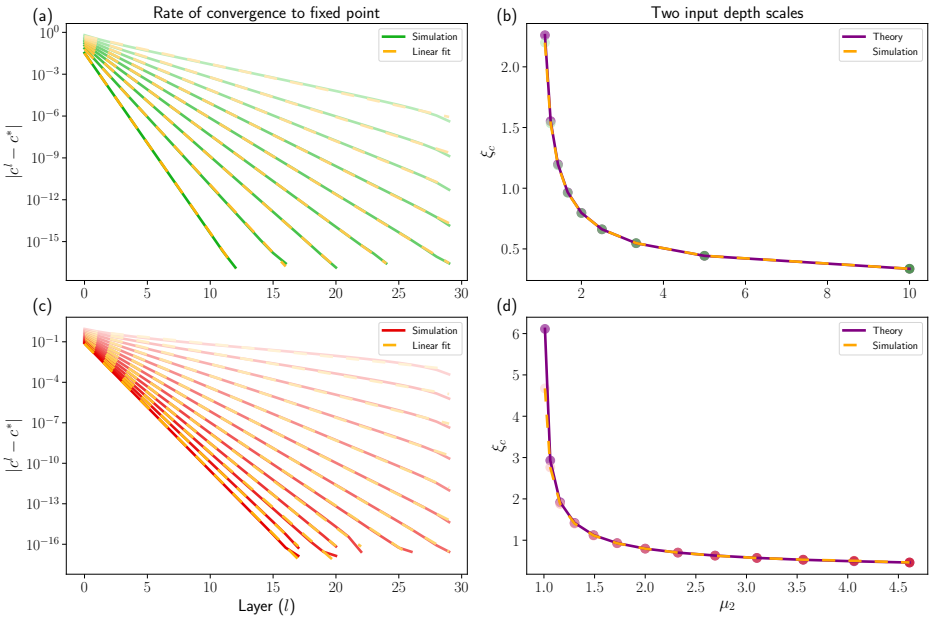


Figure 5: *Noise dependent depth scales for training.* **(a)**: Linear fits (dashed lines) to $|c^l - c^*|$ as a function of depth on a log-scale (solid lines) for varying amounts of dropout ($p = 0.1$ to $p = 0.9$ by 0.1). **(b)**: Theoretical depth scales (solid lines) versus empirically inferred scales (dashed lines) per dropout rate. Scales are inferred noting that if $|c^l - c^*| \sim e^{-l/\xi_c}$, then a linear fit, $al + b$, in the logarithmic domain gives $\xi_c \approx -\frac{1}{a}$, for large l. In other words, the negative inverse slope of a linear fit to the log differences in correlation should match the theoretical values for ξ_c. Therefore, we compare $\xi_c = -1/\ln\left[\chi(c^*)\right]$ to $-\frac{1}{a}$ for different levels of noise. **(c) - (d)**: Similar to (a) and (b), but for Gaussian noise ($\sigma_\epsilon = 0.1$ to $\sigma_\epsilon = 1.9$ by 0.15).

We next briefly discuss error signal propagation during the backward pass for noise regularised ReLU networks. When critically initialised, the error variance recurrence relation in (4) for these networks is (see Section B.4 in supplementary material)

$$\tilde{q}_\delta^l = \tilde{q}_\delta^{l+1} \frac{D_{l+1}}{D_l \mu_2}, \tag{10}$$

with the covariance between error signals in (5), given by (see Section B.5 in supplementary material)

$$\tilde{q}_{ab,\delta}^l = \tilde{q}_{ab,\delta}^{l+1} \frac{D_{l+1}}{D_{l+2}} \chi(c^*). \tag{11}$$

Note the explicit dependence on the width of the layers of the network in (10) and (11). We first consider constant width networks, where $D_{l+1} = D_l$, for all $l = 1, ..., L$. For any amount of multiplicative noise, $\mu_2 > 1$, and we see from (10) that gradients will tend to vanish for large depths. Furthermore, Figure 4 (c) plots $\chi(c^*)$ as a function of μ_2. As μ_2 increases from one, $\chi(c^*)$ decreases from one. Therefore, from (11), we also find that error signals from different inputs will tend to decorrelate at large depths.

Interestingly, for non-constant width networks, stable gradient information propagation may still be possible. If the network architecture adapts to the amount of noise being injected by having the widths of the layers grow as $D_{l+1} = D_l \mu_2$, then (10) should be at its fixed point solution. For example, in the case of dropout $D_{l+1} = D_l / p$, which implies that for any $p < 1$, each successive layer in the network needs to grow in width by a factor of $1/p$ to promote stable gradient flow. Similarly, for multiplicative Gaussian noise, $D_{l+1} = D_l(\sigma_\epsilon^2 + 1)$, which requires the network to grow in width unless $\sigma_\epsilon^2 = 0$. Similarly, if $D_{l+2} = D_{l+1}\chi(c^*) = D_l \mu_2 \chi(c^*)$ in (11), the covariance of the error signal should be preserved during the backward pass, for arbitrary values of μ_2 and $\chi(c^*)$.

Stellenbosch University https://scholar.sun.ac.za

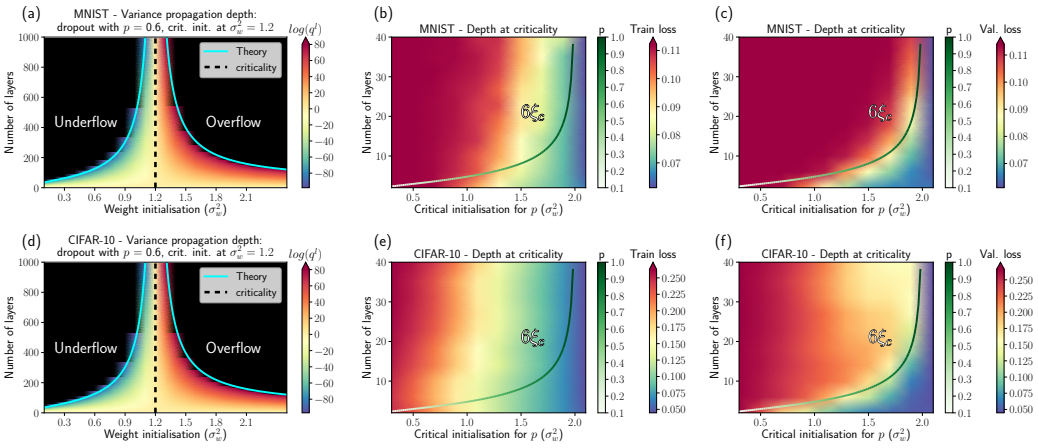


Figure 6: *Depth scale experiments on MNIST and CIFAR-10.* **(a)** Variance propagation dynamics for MNIST on and off the critical point initialisation (dashed black line) with dropout ($p = 0.6$). The cyan curve represents the theoretical boundary at which numerical instability issues are predicted to occur and is computed as $L^* = \ln(K)/\ln(\frac{\sigma_w^2}{2}\mu_2)$, where K is the largest (or smallest) positive number representable by the computer. Specifically, we use 32-bit floating point numbers and set $K = 3.4028235 \times 10^{38}$, if $\sigma_w^2 > \frac{2}{\mu_2}$ and $K = 1.1754944 \times 10^{-38}$, if $\sigma_w^2 < \frac{2}{\mu_2}$. **(b)** Depth scales fit to the training loss on MNIST for networks initialised at criticality for dropout rates $p = 0.1$ (severe dropout) to $p = 1$ (no dropout). **(c)** Depth scales fit to the validation loss on MNIST. **(d) - (f)**: Similar to (a) - (c), but for CIFAR-10. For each plot we highlight trends by smoothing the colour grid (for non smoothed versions see Section C.5 in the supplementary material).

4 Experimental results

From our analysis of deep noisy ReLU networks in the previous section, we expect that a necessary condition for such a network to be trainable, is that the network be initialised at criticality. However, whether the layer widths are varied or not for the sake of backpropagation, the correlation dynamics in the forward pass may still limit the depth at which these networks perform well.

We therefore investigate the performance of noise-regularised deep ReLU networks on real-world data. First, we validate the derived critical initialisation. As the depth of the network increases, any initialisation strategy that does not factor in the effects of noise, will cause the forward propagating signal to become increasingly unstable. For very deep networks, this might cause the signal to either explode or vanish, even within the first forward pass, making the network untrainable. To test this, we sent inputs from MNIST and CIFAR-10 through ReLU networks using dropout (with $p = 0.6$) at varying depths and for different initialisations of the network. Figure 6 (a) and (d) shows the evolution of the input statistics as the input propagates through each network for the different data sets. For initialisations not at criticality, the variance grows or shrinks rapidly to the point of causing numerical overflow or underflow (indicated by black regions). For deep networks, this can happen well before any signal is able to reach the output layer. In contrast, initialising at criticality (as shown by the dashed black line), allows for the signal to propagate reliably even at very large depths. Furthermore, given the floating point precision, if $\sigma_w^2 \neq \frac{2}{\mu_2}$, we can predict the depth at which numerical overflow (or underflow) will occur by solving for L^* in $K = \left(\sigma_w^2\mu_2/2\right)^{L^*} q^0$, where K is the largest (or smallest) positive number representable by the computer (see Section C.4 in supplementary material). These predictions are shown by the cyan line and provide a good fit to the empirical limiting depth from numerical instability.

We now turn to the issue of limited trainability. Due to the loss of correlation information between inputs as a function of noise and network depth, we expect noisy ReLU networks not to be able to perform well beyond certain depths. We investigated depth scales for ReLU networks with dropout initialised at criticality: we trained 100 networks on MNIST and CIFAR-10 for 200 epochs using SGD and a learning rate of 10^{-3} with dropout rates ranging from 0.1 to 1 for varying depths. The results

Stellenbosch University https://scholar.sun.ac.za

are shown in Figure 6 (see Section C.5 of the supplementary material for additional experimental results). For each network configuration and noise level, the critical initialisation $\sigma_w^2 = \frac{2}{\mu_2}$ was used. We indeed observe a relationship between depth and noise on the loss of a network, even at criticality. Interestingly, the line $6\xi_c$ (Schoenholz et al., 2017), seems to track the depth beyond which the relative performance on the validation loss becomes poor, more so than on the training loss. However, in both cases, we find that even modest amounts of noise can limit performance.

5 Discussion

By developing a general framework to study signal propagation in noisy neural networks, we were able to show how different stochastic regularisation strategies may impact the flow of information in a deep network. Focusing specifically on ReLU networks, we derived novel critical initialisation strategies for multiplicative noise distributions and showed that no such critical initialisations exist for commonly used additive noise distributions. At criticality however, our theory predicts that the statistics of the input should remain within a stable range during the forward pass and enable reliable signal propagation for noise regularised deep ReLU networks. We verified these predictions by comparing them with numerical simulations as well as experiments on MNIST and CIFAR-10 using dropout and found good agreement.

Interestingly, we note that a dropout rate of $p = 0.5$ has often been found to work well for ReLU networks (Srivastava et al., 2014). The critical initialisation corresponding to this rate is $(\sigma_w, \sigma_b) = (\sqrt{2p}, 0) = (1, 0)$. This is exactly the "Xavier" initialisation proposed by Glorot and Bengio (2010), which prior to the development of the "He" initialisation, was often used in combination with dropout (Simonyan and Zisserman, 2014). This could therefore help to explain the initial success associated with this specific dropout rate. Similarly, Srivastava et al. (2014) reported that adding multiplicative Gaussian noise where $\epsilon \sim \mathcal{N}(1, \sigma_\epsilon^2)$, with $\sigma_\epsilon^2 = 1$, also seemed to perform well, for which the critical initialisation is $\left(\sqrt{\frac{2}{\sigma_\epsilon^2+1}}, 0\right) = (1, 0)$, again corresponding to the "Xavier" method.

Although our initialisations ensure that individual input statistics are preserved, we further analysed the correlation dynamics between inputs and found the following: at large depths inputs become predictably correlated with each other based on the amount of noise injected into the network. As a consequence, the representations for different inputs to a deep network may become indistinguishable from each other in the later layers of the network. This can make training infeasible for noisy ReLU networks of a certain depth and depends on the amount of noise regularisation being applied.

We now note the following shortcomings of our work: firstly, our findings only apply to fully connected feed-forward neural networks and focus almost exclusively on the ReLU activation function. Furthermore, we limit the scope of our architectural design to a recursive application of a dense layer followed by a noise layer, whereas in practice a larger mix of layers is usually required to solve a specific task.

Ultimately, we are interested in reducing the number of decisions that need to made when designing deep neural networks and understanding the implications of those decisions on network behaviour and performance. Any machine learning engineer exploring a neural network based solution to a practical problem will be faced with a large number of possible design decisions. All these decisions cost valuable time to explore. In this work, we hope to have at least provided some guidance in this regard, specifically when choosing between different initialisation strategies for noise regularised ReLU networks and understanding their associated implications.

Acknowledgements

We would like to thank the reviewers for their insightful comments which improved the quality of this work. Furthermore, we would like to thank Google, the CSIR/SU Centre for Artificial Intelligence Research (CAIR) as well as the Science Faculty and the Postgraduate and International Office of Stellenbosch University for financial support. Finally, we gratefully acknowledge the support of NVIDIA Corporation with the donation of a Titan Xp GPU used for this research.

Stellenbosch University https://scholar.sun.ac.za

References

X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *Proceedings of the International Conference on Learning Representations*, 2014.

D. Sussillo and L. Abbott, "Random walk initialization for training very deep feedforward networks," *arXiv preprint arXiv:1412.6558*, 2014.

K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

D. Mishkin and J. Matas, "All you need is a good init," *Proceedings of International Conference on Learning Representations*, 2016.

X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.

N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 8609–8613.

B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli, "Exponential expressivity in deep neural networks through transient chaos," in *Advances in Neural Information Processing Systems*, 2016, pp. 3360–3368.

S. S. Schoenholz, J. Gilmer, S. Ganguli, and J. Sohl-Dickstein, "Deep information propagation," *Proceedings of the International Conference on Learning Representations*, 2017.

G. Yang and S. Schoenholz, "Mean field residual networks: On the edge of chaos," in *Advances in Neural Information Processing Systems*, 2017, pp. 7103–7114.

L. Xiao, Y. Bahri, J. Sohl-Dickstein, S. S. Schoenholz, and J. Pennington, "Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks," *Proceedings of the International Conference on Machine Learning*, 2018.

M. Chen, J. Pennington, and S. S. Schoenholz, "Dynamical isometry and a mean field theory of RNNs: Gating enables signal propagation in recurrent neural networks," *Proceedings of the International Conference on Machine Learning*, 2018.

S. Hayou, A. Doucet, and J. Rousseau, "On the selection of initialization and activation function for deep neural networks," *arXiv preprint arXiv:1805.08266*, 2018.

K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

Stellenbosch University https://scholar.sun.ac.za

Supplementary Material

In this section, we provide additional details of derivations and experimental results presented in the paper.

A Signal propagation in noise regularised neural networks

To review, given an input $\mathbf{x}^0 \in \mathbb{R}^{D_0}$, we consider the following noisy random network model

$$\tilde{\mathbf{h}}^l = W^l(\mathbf{x}^{l-1} \odot \epsilon^{l-1}) + \mathbf{b}^l, \qquad \text{for } l = 1, ..., L \tag{12}$$

where we inject noise into the model using the operator $\odot$ to denote either addition or multiplication. The vector ϵ^l is an input noise vector, sampled from a pre-specified noise distribution. For additive noise, the distribution is assumed to be zero mean. Whereas for multiplicative noise distributions, the mean is assumed to be equal to one. The weights $W^l \in \mathbb{R}^{D_l \times D_{l-1}}$ and biases $\mathbf{b}^l \in \mathbb{R}^{D_l}$ are sampled i.i.d. from zero mean Gaussian distributions with variances σ_w^2/D_{l-1} and σ_b^2, respectively, where D_l denotes the dimensionality of the l^{th} hidden layer in the network. The hidden layer activations $\mathbf{x}^l = \phi(\tilde{\mathbf{h}}^l)$ are computed element-wise using an activation function $\phi(\cdot)$, for layers $l = 1, ..., L$.

A.1 Single input signal propagation

We consider the network's behavior at initialisation. In this setting, the expected mean (over the weights, biases and noise distribution) of a unit in the pre-activations $\tilde{\mathbf{h}}_j^l$ for a single signal passing through the network will be zero with variance

$$\begin{aligned}
\tilde{q}^l &= \mathbb{E}_{\mathbf{w},\mathbf{b},\epsilon}[(\tilde{\mathbf{h}}_j^l)^2] \\
&= \mathbb{E}_{\mathbf{w},\epsilon}[\{\mathbf{w}^{l,j} \cdot (\mathbf{x}_j^{l-1} \odot \epsilon_j^{l-1})\}^2] + \mathbb{E}_{\mathbf{b}}[(\mathbf{b}_j^l)^2] \\
&= \sigma_w^2 \frac{1}{D_{l-1}} \sum_{j=1}^{D_{l-1}} \left[\phi(\tilde{\mathbf{h}}_j^{l-1})^2 \odot \mathbb{E}_\epsilon[(\epsilon_j^{l-1})^2]\right] + \sigma_b^2 \ ,
\end{aligned}$$

where we use $\mathbf{w}^{l,j}$ to denote the j-th row of W^l. The second last line relies on the bias distribution being zero mean, while the final step makes use of the independence between the inputs and the noise in the multiplicative case, and the noise being zero mean in the additive case. Furthermore, to ensure the expected value of the pre-activations remain unbiased, we only consider additive noise distributions with zero mean and multiplicative noise distributions with a mean equal to one. As in Poole et al. (2016), we make the self averaging assumption and consider the large layer width case where the previous layer's pre-activations are assumed to be Gaussian with zero mean and variance $\tilde{q}^{l-1}$. This gives the following noisy variance map

$$\tilde{q}^l = \sigma_w^2 \left\{\mathbb{E}_z\left[\phi\left(\sqrt{\tilde{q}^{l-1}}z\right)^2\right] \odot \mu_2^{l-1}\right\} + \sigma_b^2, \tag{13}$$

where $z \sim \mathcal{N}(0,1)$ and $\mu_2^l = \mathbb{E}_\epsilon[(\epsilon^l)^2]$ is the second moment of the noise distribution being sampled from at layer l. The initial input variance is given by $q^0 = \frac{1}{D_0}\mathbf{x}^0 \cdot \mathbf{x}^0$.

A.2 Two input signal propagation

To study the behaviour of a pair of signals, $\mathbf{x}^{0,a}$ and $\mathbf{x}^{0,b}$, passing through the network, we can compute the covariance in expectation over the noise and the parameters as

$$\begin{aligned}
\tilde{q}_{ab}^l &= \mathbb{E}_{\mathbf{w},\mathbf{b},\epsilon}[\tilde{\mathbf{h}}_j^{l,a}\tilde{\mathbf{h}}_j^{l,b}] \\
&= \mathbb{E}_{\mathbf{w},\mathbf{b},\epsilon}\left[\left(\mathbf{w}^{l,j} \cdot (\mathbf{x}_j^{l-1,a} \odot \epsilon_j^{l-1,a}) + \mathbf{b}_j^l\right)\left(\mathbf{w}^{l,j} \cdot (\mathbf{x}_j^{l-1,b} \odot \epsilon_j^{l-1,b}) + \mathbf{b}_j^l\right)\right] \\
&= \mathbb{E}_{\mathbf{w},\mathbf{b},\epsilon}\left[\left(\mathbf{w}^{l,j} \cdot (\mathbf{x}_j^{l-1,a} \odot \epsilon_j^{l-1,a})\right)\left(\mathbf{w}^{l,j} \cdot (\mathbf{x}_j^{l-1,b} \odot \epsilon_j^{l-1,b})\right)\right] \\
&\quad + \mathbb{E}_{\mathbf{w},\mathbf{b},\epsilon}\left[\left(\mathbf{w}^{l,j} \cdot (\mathbf{x}_j^{l-1,a} \odot \epsilon_j^{l-1,a})\right)\mathbf{b}_j^l\right] \\
&\quad + \mathbb{E}_{\mathbf{w},\mathbf{b},\epsilon}\left[\left(\mathbf{w}^{l,j} \cdot (\mathbf{x}_j^{l-1,b} \odot \epsilon_j^{l-1,b})\right)\mathbf{b}_j^l\right] \\
&\quad + \mathbb{E}_{\mathbf{w},\mathbf{b},\epsilon}\left[(\mathbf{b}_j^l)^2\right].
\end{aligned}$$

Stellenbosch University https://scholar.sun.ac.za

Since the noise is i.i.d and we have that $\mathbb{E}_{\mathbf{b}}[\mathbf{b}_j^l] = 0$, we find that

$$\tilde{q}_{ab}^l = \mathbb{E}_{\mathbf{w}}\left[\left(\mathbf{w}^{l,j}\cdot\mathbf{x}_j^{l-1,a}\right)\left(\mathbf{w}^{l,j}\cdot\mathbf{x}_j^{l-1,b}\right)\right] + \mathbb{E}_{\mathbf{b}}\left[(\mathbf{b}_j^l)^2\right] \tag{14}$$

$$= \sigma_w^2\frac{1}{D_{l-1}}\sum_{j=1}^{D_{l-1}}\left[\phi\left(\tilde{\mathbf{h}}_j^{l-1,a}\right)\phi\left(\tilde{\mathbf{h}}_j^{l-1,b}\right)\right] + \sigma_b^2, \tag{15}$$

which in the large width limit becomes

$$\tilde{q}_{ab}^l = \sigma_w^2\mathbb{E}_{z_1}\left[\mathbb{E}_{z_2}\left[\phi(\tilde{u}_1)\phi(\tilde{u}_2)\right]\right] + \sigma_b^2 \tag{16}$$

where $\tilde{u}_1 = \sqrt{\tilde{q}_{aa}^{l-1}}z_1$ and $\tilde{u}_2 = \sqrt{\tilde{q}_{bb}^{l-1}}\left[\tilde{c}^{l-1}z_1 + \sqrt{1-(\tilde{c}^{l-1})^2}z_2\right]$, with the correlation between inputs at layer l given by

$$\tilde{c}^l = \tilde{q}_{ab}^l/\sqrt{\tilde{q}_{aa}^l\tilde{q}_{bb}^l}. \tag{17}$$

Here, $z_i \sim \mathcal{N}(0,1)$ for $i = 1, 2$ and q_{aa}^l is the variance of $\tilde{\mathbf{h}}_j^{l,a}$.

B Signal propagation in noise regularised ReLU networks

In this section, we give additional details of theoretical results presented in the paper that were specifically derived for noisy ReLU networks.

B.1 Variance of input signals

Let $f(z) = \frac{e^{-z^2/2}}{\sqrt{2\pi}}$, then the variance map in (13) using ReLU, *i.e.* $\phi(a) = \max(0, a)$, becomes

$$\begin{aligned}
\tilde{q}^l &= \sigma_w^2\left[\int_{-\infty}^{\infty} f(z)\phi\left(\sqrt{\tilde{q}^{l-1}}z\right)^2 dz\right]\odot\mu_2 + \sigma_b^2 \\
&= \sigma_w^2\left[\int_{-\infty}^{0} f(z)\phi\left(\sqrt{\tilde{q}^{l-1}}z\right)^2 dz + \int_0^{\infty} f(z)\phi\left(\sqrt{\tilde{q}^{l-1}}z\right)^2 dz\right]\odot\mu_2 + \sigma_b^2 \\
&= \sigma_w^2\left[\tilde{q}^{l-1}\int_0^{\infty} f(z)z^2 dz\right]\odot\mu_2 + \sigma_b^2 \\
&= \sigma_w^2\left[\frac{\tilde{q}^{l-1}}{2}\odot\mu_2\right] + \sigma_b^2.
\end{aligned} \tag{18}$$

B.2 Correlation between input signals

Assuming that the variance map in (18) is at its fixed point $\tilde{q}^*$, which exits only if $\sigma_w^2 = \frac{2}{\mu_2}$, the correlation map in (16) for a noisy ReLU network is given by

$$\tilde{c}^l = \frac{2}{\mu_2\tilde{q}^*}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(z_1)f(z_2)\phi(\tilde{u}_1)\phi(\tilde{u}_2)dz_2dz_1 + \sigma_b^2 \tag{19}$$

where $\phi(a) = \max(a, 0)$, $f(z_i) = \frac{e^{-z_i^2/2}}{\sqrt{2\pi}}$, $\tilde{u}_1 = \sqrt{\tilde{q}^*}z_1$ and $\tilde{u}_2 = \sqrt{\tilde{q}^*}\left[\tilde{c}^{l-1}z_1 + \sqrt{1-(\tilde{c}^{l-1})^2}z_2\right]$. Note that

$$\tilde{u}_1\begin{cases}\geq 0, \text{if } z_1 > 0\\ < 0, \text{Otherwise}\end{cases}$$

$$\tilde{u}_2\begin{cases}\geq 0, \text{if } z_2 > \frac{-\tilde{c}^{l-1}z_1}{\sqrt{1-(\tilde{c}^{l-1})^2}}\\ < 0, \text{Otherwise}\end{cases},$$

Stellenbosch University https://scholar.sun.ac.za

therefore (19) becomes

$$\begin{aligned}\tilde{c}^l &= \frac{2}{\mu_2\tilde{q}^*}\int_0^\infty \int_{\frac{-\tilde{c}^{l-1}z_1}{\sqrt{1-(\tilde{c}^{l-1})^2}}}^\infty f(z_1)f(z_2)\tilde{u}_1\tilde{u}_2 dz_2 dz_1 + \sigma_b^2 \\ &= \frac{2}{\mu_2\tilde{q}^*}\sigma_w^2\int_0^\infty \int_{\frac{-\tilde{c}^{l-1}z_1}{\sqrt{1-(\tilde{c}^{l-1})^2}}}^\infty f(z_1)f(z_2)\sqrt{\tilde{q}^*}z_1\sqrt{\tilde{q}^*}\left[\tilde{c}^{l-1}z_1 + \sqrt{1-(\tilde{c}^{l-1})^2}z_2\right] dz_2 dz_1 + \sigma_b^2 \\ &= \frac{2\tilde{c}^{l-1}}{\mu_2}\int_0^\infty \int_{\frac{-\tilde{c}^{l-1}z_1}{\sqrt{1-(\tilde{c}^{l-1})^2}}}^\infty f(z_1)f(z_2)z_1^2 dz_2 dz_1 \\ &\qquad + \frac{2\sqrt{1-(\tilde{c}^{l-1})^2}}{\mu_2}\int_0^\infty \int_{\frac{-\tilde{c}^{l-1}z_1}{\sqrt{1-(\tilde{c}^{l-1})^2}}}^\infty f(z_1)f(z_2)z_1 z_2 dz_2 dz_1. \end{aligned} \tag{20}$$

The first term in (20) can then be written as

$$\frac{2\tilde{c}^{l-1}}{\mu_2}\left\{\int_0^\infty \int_{\frac{-\tilde{c}^{l-1}z_1}{\sqrt{1-(\tilde{c}^{l-1})^2}}}^0 f(z_1)f(z_2)z_1^2 dz_2 dz_1 + \int_0^\infty\int_0^\infty f(z_1)f(z_2)z_1^2 dz_2 dz_1\right\}. \tag{21}$$

In (21), the first term inside the braces is given by

$$\begin{aligned}\int_0^\infty \int_{\frac{-\tilde{c}^{l-1}z_1}{\sqrt{1-(\tilde{c}^{l-1})^2}}}^0 f(z_1)f(z_2)z_1^2 dz_2 dz_1 &= \frac{1}{2}\int_0^\infty f(z_1)z_1^2 \text{erf}\left(\frac{\tilde{c}^{l-1}z_1}{\sqrt{1-(\tilde{c}^{l-1})}}\right) dz_1 \\ &= \frac{1}{2\pi}\left[\tilde{c}^{l-1}\sqrt{1-(\tilde{c}^{l-1})^2} + \tan^{-1}\left(\frac{\tilde{c}^{l-1}}{\sqrt{1-(\tilde{c}^{l-1})^2}}\right)\right] \\ &= \frac{1}{2\pi}\left[\tilde{c}^{l-1}\sqrt{1-(\tilde{c}^{l-1})^2} + \sin^{-1}\left(\tilde{c}^{l-1}\right)\right] \end{aligned} \tag{22}$$

with $\text{erf}(a) = \frac{1}{\pi}\int_{-a}^{a} e^{-t^2}dt$. The second term inside the braces in (21) equals

$$\begin{aligned}\int_0^\infty\int_0^\infty f(z_1)f(z_2)z_1^2 dz_2 dz_1 &= \frac{1}{2}\int_0^\infty f(z_1)z_1^2 dz_1 \\ &= \frac{1}{4}. \end{aligned} \tag{23}$$

Therfore, (21) becomes

$$\frac{(\tilde{c}^{l-1})^2}{\mu_2\pi}\sqrt{1-(\tilde{c}^{l-1})^2} + \frac{\tilde{c}^{l-1}}{\mu_2\pi}\sin^{-1}\left(\tilde{c}^{l-1}\right) + \frac{\tilde{c}^{l-1}}{2\mu_2} \tag{24}$$

Similarly, the second term in (20) can be split up as follows

$$\frac{2\sqrt{1-(\tilde{c}^{l-1})^2}}{\mu_2}\left\{\int_0^\infty \int_{\frac{-\tilde{c}^{l-1}z_1}{\sqrt{1-(\tilde{c}^{l-1})^2}}}^0 f(z_1)f(z_2)z_1 z_2 dz_2 dz_1 + \int_0^\infty\int_0^\infty f(z_1)f(z_2)z_1 z_2 dz_2 dz_1\right\}. \tag{25}$$

The first term inside the braces of (25) is

$$\begin{aligned}\int_0^\infty \int_{\frac{-\tilde{c}^{l-1}z_1}{\sqrt{1-(\tilde{c}^{l-1})^2}}}^0 f(z_1)f(z_2)z_1 z_2 dz_2 dz_1 &= \frac{1}{\sqrt{2\pi}}\int_0^\infty f(z_1)z_1\left[e^{-\frac{\tilde{c}^{l-1}z_1^2}{2(1-(\tilde{c}^{l-1})^2)}} - 1\right] dz_1 \\ &= \frac{1}{\sqrt{2\pi}}\left\{\frac{1-(\tilde{c}^{l-1})^2}{\sqrt{2\pi}} - \frac{1}{\sqrt{2\pi}}\right\} \\ &= -\frac{(\tilde{c}^{l-1})^2}{2\pi} \end{aligned} \tag{26}$$

Stellenbosch University https://scholar.sun.ac.za

and the second term is

$$\int_0^\infty \int_0^\infty f(z_1)f(z_2)z_1 z_2 dz_2 dz_1 = \frac{1}{\sqrt{2\pi}} \int_0^\infty f(z_1) z_1 dz_1 = \frac{1}{2\pi}. \tag{27}$$

Putting these two terms together, (25) becomes

$$-\frac{(\tilde{c}^{l-1})^2}{\mu_2 \pi}\sqrt{1-(\tilde{c}^{l-1})^2} + \frac{1}{\mu_2 \pi}\sqrt{1-(\tilde{c}^{l-1})^2}. \tag{28}$$

Finally, summing all the terms in (24) and (28) gives (19) as

$$\tilde{c}^l = \frac{1}{\mu_2}\left\{\frac{\tilde{c}^{l-1}\sin^{-1}\left(\tilde{c}^{l-1}\right) + \sqrt{1-(\tilde{c}^{l-1})^2}}{\pi} + \frac{\tilde{c}^{l-1}}{2}\right\}. \tag{29}$$

We note that for the noiseless case, (29) is identical to the result recently obtained by Hayou et al. (2018), where the authors used a slightly different approach.

B.3 Depth scales for trainability

We recap the result in Schoenholz et al. (2017) and adapt the derivation for the specific case of a noisy ReLU network. Let $c^l = c^* + \varepsilon^l$, such that as long as $\lim_{l\to\infty} c^l = c^*$ exist we have that $\varepsilon \to 0$ as $l \to \infty$. Then Schoenholz et al. (2017) derived the following asymptotic recurrence relation

$$\varepsilon^{l+1} = \varepsilon^l \chi(c^*) + \mathcal{O}((\varepsilon^l)^2), \tag{30}$$

where

$$\chi(c^*) = \sigma_w^2 \mathbb{E}_{z_1}\left[\mathbb{E}_{z_2}\left[\phi'(\tilde{u}_1^*)\phi'(\tilde{u}_2^*)\right]\right], \tag{31}$$

with $\tilde{u}_1^* = \tilde{u}_1 = \sqrt{\tilde{q}^*} z_1$ and $\tilde{u}_2^* = \sqrt{\tilde{q}^*}\left[\tilde{c}^* z_1 + \sqrt{1-(\tilde{c}^*)^2} z_2\right]$. Now, specifically for a noisy ReLU network where $\sigma_w^2 = \frac{2}{\mu_2}$, we have that

$$\begin{aligned}
\chi(c^*) &= \frac{2}{\mu_2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(z_1)f(z_2)\phi'(\tilde{u}_1^*)\phi'(\tilde{u}_2^*)dz_2 dz_1 \\
&= \frac{2}{\mu_2}\int_0^\infty \int_{-\frac{c^* z_1}{\sqrt{1-(c^*)^2}}}^{\infty} f(z_1)f(z_2)dz_2 dz_1 \\
&= \frac{2}{\mu_2}\int_0^\infty f(z_1)\frac{1}{2}\left[\text{erf}\left(\frac{c^* z_1}{\sqrt{2}\sqrt{1-(c^*)^2}}\right)+1\right]dz_1 \\
&= \frac{2}{\mu_2}\left[\frac{1}{2\pi}\tan^{-1}\left(\frac{c^*}{\sqrt{1-(c^*)^2}}\right)+\frac{1}{4}\right] \\
&= \frac{1}{\mu_2 \pi}\left[\sin^{-1}(c^*)+\frac{\pi}{2}\right]
\end{aligned} \tag{32}$$

Note that $\chi(c^*)$ is a constant, thus for large l the solution to the recurrence relation in (30) is expected to be exponential, *i.e.* $\varepsilon^l \sim e^{-l/\xi_c}$. Here ξ_c, is considered the *depth scale*, which controls how deep discriminatory information about the inputs can propagate through the network. We can then solve for ξ_c to find

$$\xi_c = -1/\ln(\chi(c^*)) = -\ln\left[\frac{\sin^{-1}(c^*)}{\mu_2\pi} + \frac{1}{2\mu_2}\right]^{-1}. \tag{33}$$

Stellenbosch University https://scholar.sun.ac.za

B.4 Variance of error signals

Under the mean field assumption, Schoenholz et al. (2017) approximates the error signal at layer l by a zero mean Gaussian with variance

$$\tilde{q}_\delta^l = \tilde{q}_\delta^{l+1} \frac{D_{l+1}}{D_l} \sigma_w^2 \mathbb{E}_z \left[\phi' \left(\sqrt{\tilde{q}^l} z \right)^2 \right], \tag{34}$$

where $\tilde{q}_\delta^l = \mathbb{E}[(\tilde{\delta}_i^l)^2]$, with $\tilde{\delta}_i^l = \phi'(\tilde{\mathbf{h}}_i^l) \sum_{j=1}^{D_{l+1}} \tilde{\delta}_j^{l+1} W_{ji}^{l+1}$. In our context, for a critically initialised noisy ReLU network we have that

$$\tilde{q}_\delta^l = \tilde{q}_\delta^{l+1} \frac{D_{l+1}}{D_l} \frac{2}{\mu_2} \int_0^\infty f(z) dz \tag{35}$$

$$= \tilde{q}_\delta^{l+1} \frac{D_{l+1}}{D_l} \frac{1}{\mu_2}. \tag{36}$$

B.5 Correlation between error signals

The covariance between error signals is approximated using

$$\tilde{q}_{ab,\delta}^l = \tilde{q}_{ab,\delta}^{l+1} \frac{D_{l+1}}{D_{l+2}} \sigma_w^2 \mathbb{E}_{z_1} \left[\mathbb{E}_{z_2} \left[\phi'(\tilde{u}_1) \phi'(\tilde{u}_2) \right] \right], \tag{37}$$

where $\tilde{u}_1$ and $\tilde{u}_2$ are defined as was done in the forward pass. Here, we simply use the result in (32) for noisy ReLU networks to find

$$\tilde{q}_{ab,\delta}^l = \tilde{q}_{ab,\delta}^{l+1} \frac{D_{l+1}}{D_{l+2}} \chi(c^*) \tag{38}$$

$$= \tilde{q}_{ab,\delta}^{l+1} \frac{D_{l+1} \left[\sin^{-1}(c^*) + \frac{\pi}{2} \right]}{D_{l+2} \mu_2 \pi}. \tag{39}$$

C Experimental details

In this section we provide additional details regarding our experiments in the paper. Code to reproduce all the experiments is available at https://github.com/ElanVB/noisy_signal_prop.

C.1 Input data

For all experiments the network input data properties that remain consistent (unless stated otherwise) are as follows: each observation consists of 1000 features and each feature value is drawn i.i.d. from a standard normal distribution.

C.2 Variance propagation dynamics

The experiments conducted to gather results for Figures 2 and 3 aim to empirically show the relationship between the variances at arbitrary layers in a neural network.

Iterative map: For the results depicted in Figures 2 (a) and 3 (a), the experimental set up is as follows. The data used as input to these experiments comprises of 30 sets of 30 observations. The input is scaled such that the variance of observations within each set is the same and the variance across each set is different and forms a range of $q_{\text{set}} \in [0, 15]$. As such, our results are averaged over 30 observations and 50 samplings of initial weights to a single hidden-layer network.

Convergence dynamics: For the results depicted in Figures 2 (b) and 3 (b), the experimental set up is as follows. The data used as input to these experiments comprises of a set of 50 observations scaled such that each observation's variance is four ($q = 4$). As such, our results are averaged over 50 observations and 50 samplings of initial weights to a 15 hidden-layer network.

C.3 Correlation propagation dynamics

The experiments conducted to gather results for Figure 4 and 5 aim to empirically show the relationship between the correlations of observations at arbitrary layers in a neural network.

Stellenbosch University https://scholar.sun.ac.za

Iterative map: For the results depicted in Figure 4 (a), the experimental set up is as follows. The data used as input to these experiments comprises of 50 sets of 50 observations. The first observation in each set is sampled from a standard normal distribution and subsequent observations are generated such that the correlation between the first element and the i^{th} element form a range of $\text{corr}_{0,i} \in [0, 1]$. As such, our results are averaged over 50 observations and 50 samplings of initial weights to a single hidden-layer network.

Convergence dynamics: For the results depicted in Figure 4 (b), the experimental set up is as follows. The data used as input to these experiments comprises of three sets of 50 equally correlated observations. Each set has a different correlation value such that $\text{corr}_{\text{set}} \in \{0, 0.5, 0.9\}$. As such, our results are averaged over 50 observations and 50 samplings of initial weights to a 15 hidden-layer network.

Confirmation of exponential rate of convergence for correlations: This section discusses how the results depicted in Figure 5 are acquired. These experiments support the assumption that the rate of convergence for correlations is exponential when using noise regularisation with rectifier neural networks. The experimental set up for this section is very similar to that of the above convergence dynamics experiment, the only difference being the statistics we calculate from the correlation values. The aspect of this experiment that may seem the most unclear is the reason why we claim that the negative inverse slope of a linear fit to the $\log$ differences in correlation should match the theoretical values for ξ_c. The derivation to justify this is as follows. If a good fit of the form $al + b$ can be found in the logarithmic domain for the rate of convergence, it would strongly indicate that the convergence rate is exponential. Following this, we set the problem up like so:

$$\begin{aligned} |c^l - c^*| &\approx e^{-l/\xi_c} \\ \therefore \ln\left(|c^l - c^*|\right) &\approx \frac{-l}{\xi_c}. \end{aligned}$$

Let us now assume that $\ln\left(|c^l - c^*|\right)$ can be linearly approximated:

$$\begin{aligned} \therefore \ln\left(|c^l - c^*|\right) &\approx al + b, \\ \therefore al + b &\approx \frac{-l}{\xi_c}, \\ \therefore \xi_c &\approx \frac{-l}{al + b}. \end{aligned}$$

Since we are concerned with deep neural networks, we can take the limit as l becomes arbitrarily large and see that as l grows the effect of b decreases ($\lim_{l\to\infty} |al| \gg |b|$). Thus, we continue like so:

$$\begin{aligned} \lim_{l\to\infty} \xi_c &\approx \lim_{l\to\infty} \frac{-l}{al} \\ &\approx -\frac{1}{a}. \end{aligned}$$

Thus, we have come to the finding that if the correlation rate of convergence is exponential and we work with deep neural networks, the negative inverse slope of a linear fit to the $\log$ differences in correlation should match the theoretical values for ξ_c. Figure 5 shows that the theory closely matches this approximation.

C.4 Depth scales

This section handles the experiments conducted related to determining the maximum depth variance information can stably propagate through a network and the depth at which these networks can be trained, both depicted in Figure 6.

The MNIST and CIFAR-10 datasets were used and were pre-processed using standard techniques. Throughout these experiments mini-batches of 128 observations were used.

Variance depth scales: The experiments depicted in Figures 6 (a) and (d) are interested in testing the numerical stability of networks initialised using different σ_w^2 values while using 32-bit floating point

Stellenbosch University https://scholar.sun.ac.za

numbers. To test the depth of stable variance propagation, a network with 1000 hidden layers is used. The network used in this experiment makes use of dropout with $p = 0.6$, where p is the probability of keeping a neuron's value, thus the critical value for σ_w^2 is 1.2. As such, a linearly spaced range of $\sigma_w^2 \in [0.1, 2.5]$ is used to select 25 different values.

We use the following approach to predict the depth beyond which variances become numerically unstable. At criticality for multiplicative noise $(\sigma_w, \sigma_b) = (\sqrt{2/\mu_2}, 0)$, however, for weights initialised off this critical point (18) becomes

$$\begin{aligned}\tilde{q}^l &= \tilde{q}^{l-1}\left(\frac{\sigma_w^2\mu_2}{2}\right) \\ &= \left[\tilde{q}^{l-2}\left(\frac{\sigma_w^2\mu_2}{2}\right)\right]\left(\frac{\sigma_w^2\mu_2}{2}\right) \\ &= \tilde{q}^0\left(\frac{\sigma_w^2\mu_2}{2}\right)^l. \end{aligned} \tag{40}$$

If $\sigma_w^2 > \frac{2}{\mu_2}$, we let $\tilde{q}^l = K$, where K is the largest positive number representable by the computer. In our case, using 32-bit floating point precision, this number is equal to 3.4028235×10^{38}. Otherwise, if $\sigma_w^2 < \frac{2}{\mu_2}$ we select $K = 1.1754944 \times 10^{-38}$, the smallest possible positive number. Furthermore, let L^* represent the layer l in (40) at which the value K is reached, then we can scale our input data such that $\tilde{q}^0 = 1$ and solve for L^* to find

$$L^* = \ln(K)/\ln\left(\frac{\sigma_w^2\mu_2}{2}\right). \tag{41}$$

Therefore, we expect numerical instability issues to occur beyond a depth of L^*.

Trainable depth scales: The experiments depicted in Figures 6 (b), (c), (e) and (f) are concerned with determining at what depth a critically initialised network with a specified dropout rate can train effectively. To this end, 10 linearly spaced values for dropout on the range $p \in [0.1, 1.0]$ and 10 linearly spaced network depths on the integer range $l \in [2, 40]$ are tested.

The task presented to the network in this experiment is to learn the identity function within 200 epochs. As such, the network is set up as an auto-encoder and uses stochastic gradient decent with a learning rate of 10^{-3}. The input data is divided into a training and validation set, each containing 50000 and 10000 observations respectively.

C.5 Additional results

In this section we provide some additional experiments on the training dynamics of deep noisy ReLU networks from different initialisations.

In Figure 7 we compare the standard "He" initialisation (blue) with the critical initialisation (green) for a ReLU network with dropout regularisation $(p = 0.8)$. By not initialising at criticality due to dropout noise, the variance map for the "He" strategy no longer lies on the identity line in (a) and as a result, the forward propagating signal can be seen to explode in (b). However, by compensating for the amount of injected noise, the above derived critical initialisation for dropout preserves the signal throughout the entire forward pass, with roughly constant variance dynamics.

Next, we provide some additional experiments on the trainability of deep ReLU networks with dropout on real-world data sets.

From our analysis in the paper, we expect that as the depth of the network increases, any initialisation strategy that does not factor in the effects of noise, will cause the forward propagating signal to become increasingly unstable. For very deep networks, this might cause the signal to either explode or vanish, even within the first forward pass, making the network untrainable.

To test this, we trained a denoising autoencoder network with dropout noise ($p = 0.6$) on MNIST and CIFAR-10 using squared reconstruction loss. We consider several network depths ($L = 30, 100, 200$), learning rates ($\alpha = 0.1, 0.01, 0.001, 0.0001$) and optimisation procedures (SGD and Adam), with 1000 neurons in each layer. The results for training on CIFAR-10 are shown in Figure 8 for both the "He" intialisation (blue) and the critical dropout initialisation (green). (For MNIST, see Figure 9; the

Stellenbosch University https://scholar.sun.ac.za

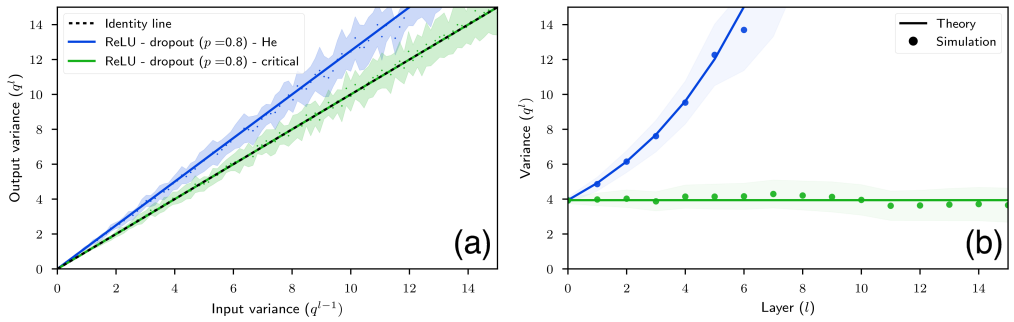


Figure 7: *Critical initialisation for ReLU networks with dropout.* Lines correspond to theoretical predictions and points to numerical simulations, for random ReLU networks with dropout ($p = 0.8$), initialised according to the method proposed by He et al. (2015) (blue) and at criticality (green). **(a)**: Iterative variance map where the identity line is displayed as a dashed black line. **(b)**: Variance dynamics during forward signal propagation.

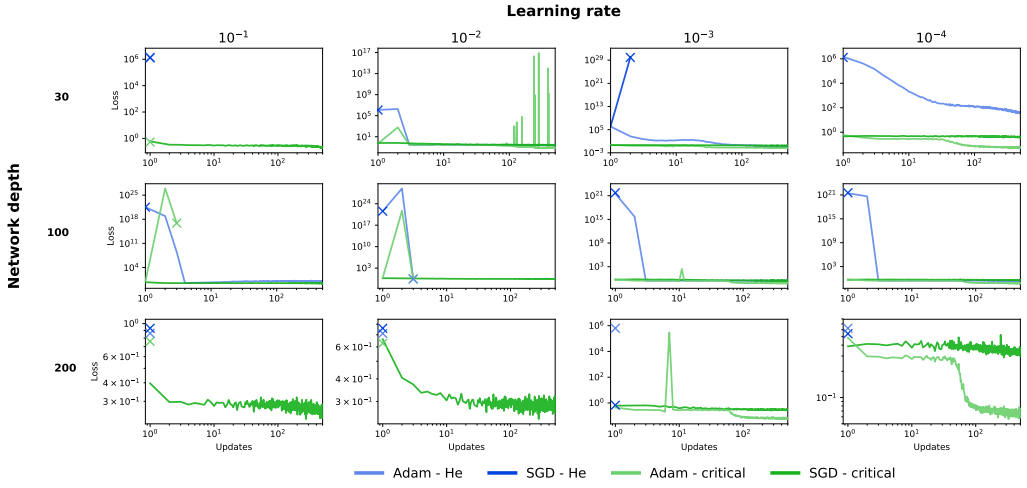


Figure 8: Comparing the "He" initialisation strategy to critical dropout initialisation for ReLU networks using dropout $(p = 0.6)$ on CIFAR-10. While networks initialised at criticality (green) are able to train at large depths $(L = 200)$ as seen in the bottom row, networks initialised with the "He" strategy (blue) become untrainable irrespective of the chosen learning rate or optimisation procedure. An "X" marks the point at which a network completely stopped training. Training losses and number of network updates are shown in log-scale.

core trends and resulting conclusions regarding network trainability is the same for both data sets, which we discuss below.)

As the depth increases, moving from the top to the bottom row in Figure 8, networks initialised at the critical point for dropout seem to remain trainable even up to a depth of 200 layers (we see the loss start to decrease over five epochs). In contrast, networks using the "He" initialisation become increasingly more difficult to train, with no training taking place at very large depths. These findings make sense in terms of the variance dynamics analysed in the paper, however, these experimental successes seem to run counter to our theoretical analysis of trainable depth scales (this contradiction can also be seen in Figure 6). Understanding this discrepancy is of particular interest to us.

To verify that the lack of training in Figure 8 is due to poor signal propagation, we plot the empirical variance of the pre-activations in Figure 10, for the first forward pass of a 200 layer autoencoder

Stellenbosch University https://scholar.sun.ac.za

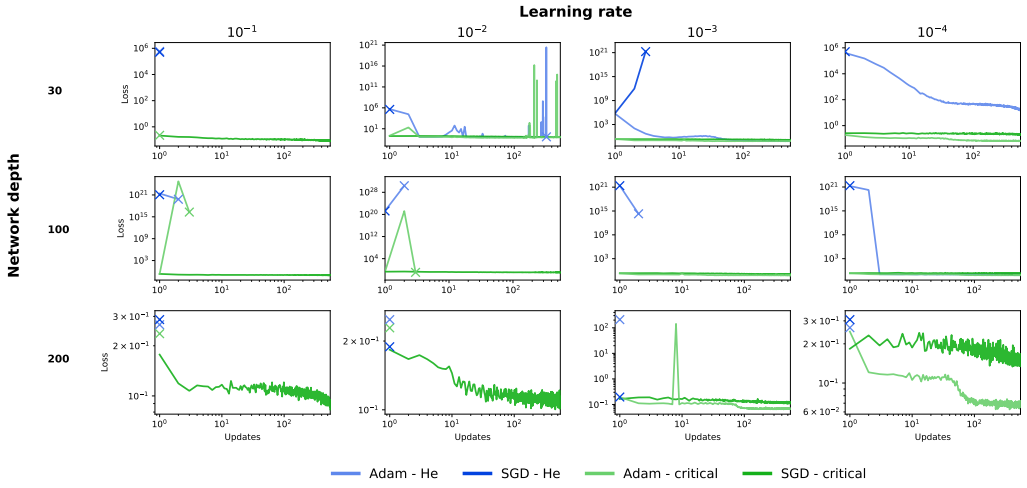


Figure 9: Comparing the "He" initialisation strategy to critical dropout initialisation for ReLU networks using dropout ($p = 0.6$) on MNIST. While networks initialised at criticality (green) are able to train at large depths ($L = 200$) as seen in the bottom row, networks initialised with the "He" strategy (blue) become untrainable irrespective of the chosen learning rate or optimisation procedure. An "X" marks the point at which a network completely stopped training. Training losses and number of network updates are shown in log-scale.

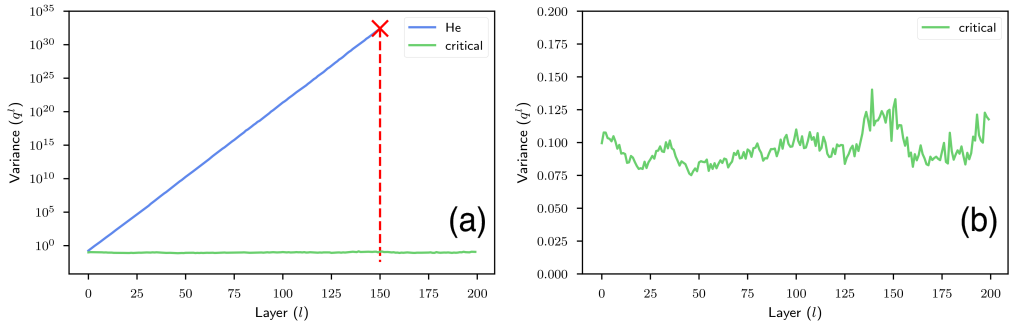


Figure 10: *Variance dynamics for signal propagation in the first forward pass for a 200 layer autoencoder network fed a batch of 500 training examples from CIFAR-10.* **(a)** Exploding activation variance (blue) reaching overflow levels (marked with a red "X") for the "He" intialisation, with no signal reaching the output layer (shown in log-scale). **(b)** Zoomed in display of the roughly constant variance dynamics in (a) for the critical dropout initialisation.

network. For the "He" initialisation, the variance in (a) grows rapidly to the point of causing numerical instability and overflow (indicated by the red dashed line), well before any signal is able to reach the output layer. However as shown in (b), by initialising at criticality, signal is able to propagate reliably even at large depths.

Stellenbosch University https://scholar.sun.ac.za

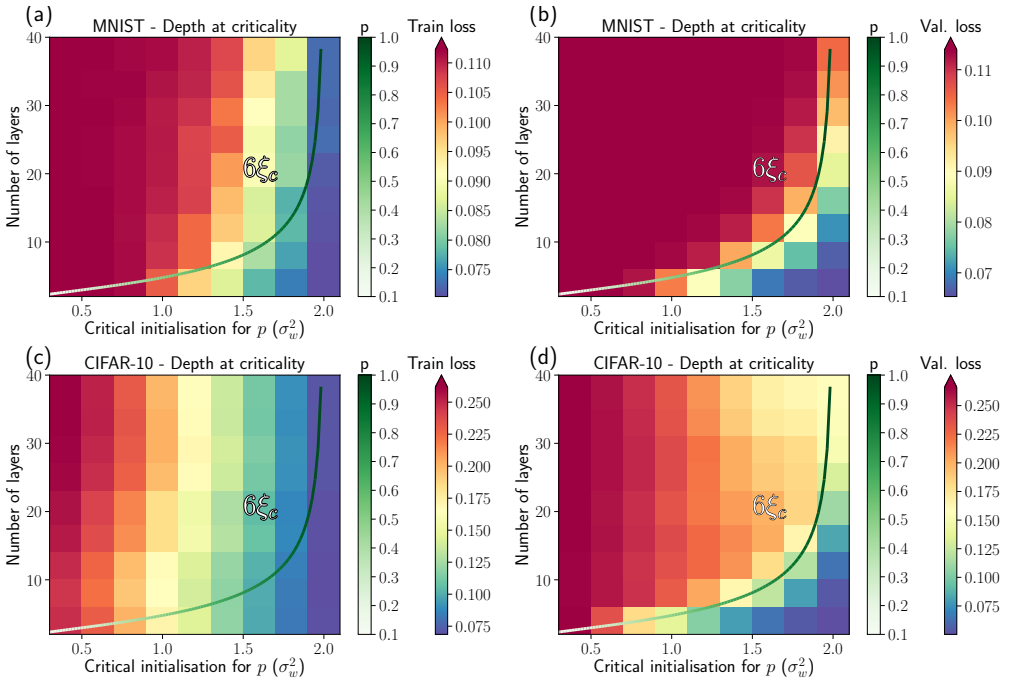


Figure 11: *Depth scale experiments on MNIST and CIFAR-10.* **(a)** Depth scales fit to the training loss on MNIST for networks initialised at criticality for dropout rates $p = 0.1$ (severe dropout) to $p = 1$ (no dropout). **(b)** Depth scales fit to the validation loss on MNIST. **(c) - (d)**: Similar to (a) - (c), but for CIFAR-10.

Stellenbosch University https://scholar.sun.ac.za

3.4 Discussion

In **this chapter**, we showed how noise regularisation affects initialisation and how both initialisation and noise interact with network depth. Specifically, in the case of multiplicative noise corruption, we derived appropriate scales for the weights of a deep neural network to ensure stable signal propagation. However, the deeper the network the more correlated the internal representations of the inputs become during the first forward pass. This has the effect of limiting the depth to which networks can be trained.

Since publication, we have become aware of previous work by Hendrycks and Gimpel (2016), who also arrived at the critical initialisation (albeit more heuristically), specifically for dropout. Compared to their work, we consider our analysis to be more general and rigorous. Our contributions also go beyond specifying an initialisation scheme by analysing the correlation dynamics and limiting depth of noise in ReLU networks.

The work presented in this chapter leads us to the following conclusion: critical initialisation is important for stable signal propagation at *large depth*, but at the same time noise regularisation results in networks only performing well at *limited depth*. Therefore, a natural question to ask is: if noise regularisation (e.g. dropout) limits depth, does critical initialisation still matter? Or can we perhaps do better? We investigate this question in the **next chapter**.

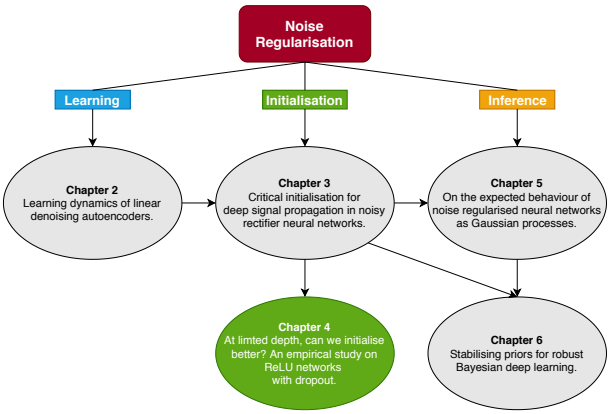
Stellenbosch University https://scholar.sun.ac.za

Chapter 4

At limited depth, can we initialise better?

Paper
Pretorius, A., van Biljon, E., van Niekerk, B., Eloff, R., Reynard, M., James, S., Rosman, B., Kamper, H., & Kroon, S. At limited depth, can we initialise better? An empirical study on ReLU networks with dropout. *Preprint (arXiv, 2019).*
Available at: https://arxiv.org/pdf/1910.05725.pdf

Code
Pretorius, A., van Biljon, E., van Niekerk, B., Eloff, R., Reynard, M. (2019). Code for: At limited depth, can we initialise better? An empirical study on ReLU networks with dropout.
Available at: https://github.com/arnupretorius/betterInitAtLimitedDepth_2019



4.1 Introduction

Critical initialisation is more crucial for neural networks of large depth. If the signal in a neural network finds itself in either the exploding or vanishing regime because of poor initialisation, increased depth will only cause further signal growth or decay. At large depth, this process can cause the network to become numerically unstable and entirely untrainable.

Shallow-to-moderately deep networks will naturally guard against numerical instability because of their shorter depth horizon for compounding effects. This means that even

Stellenbosch University https://scholar.sun.ac.za

if the network is not properly initialised, a numerically stable signal might still be able to reach the output layer to allow for training.

It is important to realise that most critical initialisation strategies, such as those derived in Chapter 3, rely on assumptions that are only valid during the first forward pass. Furthermore, these strategies seek to satisfy certain criteria that exhibit good properties at the start of training. These good properties might not persist *during* training, or they might improve training but negatively effect generalisation.

In Chapter 3, we showed that noise regularisation limits the depth to which networks can be trained. Therefore, training with noise is only possible for networks of moderate depth, where the compounding effects for unstable signal propagation will be less of an issue. If this is the case, critical initialisation becomes less important, which leads us to the following question: *is it worthwhile to explore the initialisation landscape around criticality in networks of limited depth, in the hope of finding previously undiscovered benefits for training or generalisation?*

It is conceivable that some benefits might exist around the critical initialisation. For example, the spectrum of the input-output Jacobian has been shown to be well-behaved just off of criticality (Saxe *et al.*, 2014). This behaviour in the spectrum also seems to correlate with improvements in training and generalisation (Pennington *et al.*, 2017). Still, very little work has focused on exploring the initialisation landscape around criticality.

In this chapter, we investigate the initialisation landscape around criticality for ReLU neural networks of moderate depth that use dropout regularisation. We perform a large-scale randomised controlled trial experiment in which we train thousands of networks while controlling for confounding effects. This is done by training each network over many different hyperparameters that are randomly sampled for each chosen initialisation. We explore the landscape around criticality in a principled way to ensure that each initialisation theoretically satisfies the necessary criteria for training. A statistical analysis of our findings shows that, rather surprisingly, there is no statistically significant difference in training speed and generalisation between the critical initialisation and a wide range of alternative non-critical initialisations. We highlight that these conclusions also hold for standard ReLU networks without any dropout.

4.2 Contribution statement

This work started as a two-week coding sprint initiated and managed by myself, which included Elan van Biljon, Ryan Eloff, Matthew Reynard, Benjamin van Niekerk and Steve James. The idea of conducting a large scale randomised controlled experiment to test differences in initialisation around criticality was my own, but was inspired by work of a similar nature (Novak *et al.*, 2018). Before the sprint, I wrote the base level code that was the start of the project, which included building models and running basic experiments. During the sprint, the rest of the team and I extended the code to the scale required to run all the experiments in the paper. Steve James specifically helped with setting up experiments to run on a large cluster of computers at the University of the Witswatersrand. Elan van Biljon was instrumental in refactoring some of the code to make use of multithreading, which greatly sped up our experiments. I performed the statistical analysis of our results and wrote the paper. Dr Benjamin Rosman, Dr Herman Kamper and Dr Steve Kroon as well as the rest of the team gave feedback on the draft and helped with editorial corrections.

Stellenbosch University https://scholar.sun.ac.za

If dropout limits trainable depth, does critical initialisation still matter? A large-scale statistical analysis on ReLU networks

Arnu Pretorius[1,4] **Elan van Biljon**[1] **Benjamin van Niekerk**[2]
Ryan Eloff[2] **Matthew Reynard**[2] **Steven James**[5]
Benjamin Rosman[3,5] **Herman Kamper**[2] **Steve Kroon**[1]
[1]Computer Science Division, Stellenbosch University
[2]Department of E&E Engineering, Stellenbosch University
[3]Council for Scientific and Industrial Research, South Africa
[4]CSIR/SU Centre for Artificial Intelligence Research, South Africa
[5]Department of Computer Science and Applied Mathematics, University of the Witwatersrand

Abstract

Recent work in signal propagation theory has shown that dropout limits the depth to which information can propagate through a neural network. In this paper, we investigate the effect of initialisation on training speed and generalisation for ReLU networks within this depth limit. We ask the following research question: given that critical initialisation is crucial for training at large depth, if dropout limits the depth at which networks are trainable, does initialising critically still matter? We conduct a large-scale controlled experiment, and perform a statistical analysis of over 12000 trained networks. We find that (1) trainable networks show no statistically significant difference in performance over a wide range of non-critical initialisations; (2) for initialisations that show a statistically significant difference, the net effect on performance is small; (3) only extreme initialisations (very small or very large) perform worse than criticality. These findings also apply to standard ReLU networks of moderate depth as a special case of zero dropout. Our results therefore suggest that, in the shallow-to-moderate depth setting, critical initialisation provides zero performance gains when compared to off-critical initialisations and that searching for off-critical initialisations that might improve training speed or generalisation, is likely to be a fruitless endeavour.

Preliminary work.

1 Introduction

Dropout is arguably one of the most popular and successful forms of regularisation for deep neural networks (Srivastava et al., 2014). This has sparked research into analysing dropout's effects (Wang and Manning, 2013; Wager et al., 2013; Baldi and Sadowski, 2013), extending dropout's mechanism of regularisation (Wan et al., 2013; Gal et al., 2017; Gomez et al., 2018; Ghiasi et al., 2018) and connecting dropout to different Bayesian inference methods (Kingma et al., 2015; Gal and Ghahramani, 2016; Molchanov et al., 2017). Despite its success, dropout has also been shown to limit the *trainable depth* of a neural network (Schoenholz et al., 2017).

At initialisation, the random weight projection at each layer combined with dropout may cause inputs to become uniformly correlated beyond a certain depth. Thus discriminatory information in the inputs may vanish before reaching the output layer. The trainable depth of a network is the maximum depth to which this information is able to propagate forward without completely vanishing in this way. Schoenholz et al. (2017) arrive at this result through a *mean field* analysis of dropout at initialisation.

Mean field theory provides a powerful approach to analysing deep neural networks and has become a cornerstone of recent discoveries in improved initialisation schemes. These schemes, often referred to as *critical initialisations*, ensure stable signal propagation dynamics by preserving second moment input statistics during the forward pass, even at infinite depth. Critical initialisation has made it possible to train *extremely deep* networks (sometimes up to 10000 layers) for a variety of different architectures (Pennington et al., 2017; Xiao et al., 2018; Chen et al., 2018). Using the tools of mean field theory, Pretorius et al. (2018) extend these results to fully-connected ReLU networks with

Stellenbosch University https://scholar.sun.ac.za

multiplicative noise regularisation. These results hold for a general class of noise distributions, while earlier work (Hendrycks and Gimpel, 2016) describes dropout-specific initialisation schemes.

For non-critical initialisation, signal propagation can become unstable and result in the saturation of activation functions. In the particular case of ReLU activations, numerical instability (overflow or underflow) can arise when training very deep networks. Despite this, when training ReLU networks of a finite depth, there is a range of trainable but non-critical initialisations. That is, there exists a "band" of valid initialisations around the critical point. It is conceivable that using these alternative, non-critical initialisations may confer some benefits. For example, Saxe et al. (2014) note that just off of criticality, the spectrum of the input-output Jacobian can be well behaved, which has been linked to improvements in training and generalisation (Pennington et al., 2017). This leads us to the following question.

Question: *If dropout limits the depth to which networks can train, does critical initialisation still matter?* Given that stable signal propagation at extreme depths is no longer a concern, are there alternative initialisations that might perform better than the critical initialisation?

To investigate the above research question, we conduct a large-scale randomised control trial (RCT)—an approach borrowed from the medical community—to compare training speed and generalisation for ReLU neural networks with dropout for different initialisations. We consider multiple datasets, training algorithms, dropout rates and combinations of hyperparameters to avoid confounding effects. To the best of our knowledge, this is the first application of RCTs in a deep learning context. A statistical analysis of our results leads to the following insight.

Answer: *There is no statistically significant difference between the critical initialisation and a wide neighbourhood of non-critical initialisations, as measured by training speed and generalisation.* In our experiment we find that this also applies to standard ReLU networks without dropout, for which the critical initialisation is the popular "He" initialisation (He et al., 2015). Our findings seem to indicate that networks of moderate depth (less than 20 layers) are in fact very insensitive to initialisation. In addition, we conclude that exploring the initialisation landscape around criticality in the hope of finding previously undiscovered benefits, is unlikely to be a fruitful enterprise.

2 Background

We model the expected value of a target variable $\mathbf{y}$ conditioned on an input $\mathbf{x}$, i.e. $\mathbb{E}(\mathbf{y}|\mathbf{x})$, using a fully-connected feedforward neural network with dropout. Given an input $\mathbf{x}^0 \in \mathbb{R}^{D_0}$, we can define this neural network recursively as

$$\mathbf{x}^l = \phi(\tilde{\mathbf{h}}^l), \qquad \tilde{\mathbf{h}}^l = W^l \left(\mathbf{x}^{l-1} \odot \frac{\epsilon^{l-1}}{1-\theta} \right) + \mathbf{b}^l, \quad (1)$$

for $l = 1, ..., L$, where L is the total number of hidden layers, $\odot$ denotes element-wise multiplication, and $\epsilon^l \sim \text{Bern}(1-\theta)$ is a Bernoulli noise vector, corresponding to a dropout rate θ. The dimensionality of hidden layer l is denoted as D_l, and activations at each layer are computed element-wise using $\phi(a) = \text{ReLU}(a) = \max(0, a)$. The initial weights $W^l \in \mathbb{R}^{D_l \times D_{l-1}}$ and biases $\mathbf{b}^l \in \mathbb{R}^{D_l}$ are sampled i.i.d. from zero-mean Gaussian distributions with variances σ_w^2/D_{l-1} and σ_b^2, respectively.

We focus on ReLU because of its widespread use and empirical success and consider the fully-connected setting since derived conclusions for these networks often generalise to other architectures, e.g. convolutional networks (He et al., 2015; Xiao et al., 2018). Through their work, Schoenholz et al. (2017) also hypothesise that the signal propagation behaviour of many different architectures is likely to be governed by the fully-connected case.

2.1 Mean field theory for signal propagation

Poole et al. (2016), Schoenholz et al. (2017) and Pretorius et al. (2018) use mean field theory to analyse fully-connected feedforward neural networks at initialisation. For large layer widths, each pre-activation (the linear combination of the incoming connections from the previous layer) at initialisation in any given layer of the network represents a large sum of i.i.d. random variables. According to the central limit theorem, this sum will tend to a Gaussian distribution in the limit of infinite width. Using the above observation, the mean field approach is to fit Gaussian distributions over all the pre-activation units through moment matching to describe the behaviour of wide random neural networks at initialisation.

In more detail, consider two inputs $\mathbf{x}_1^0$ and $\mathbf{x}_2^0$. Denote the scalar pre-activation at unit j in layer l for input $\mathbf{x}_1^0$ as $\tilde{h}_j^{l,1}$. For fully-connected ReLU networks with dropout, Pretorius et al. (2018) derive the joint distribution over the pre-activations in expectation over the network parameters and the noise as

$$p\left(\tilde{h}_j^{l,1}, \tilde{h}_j^{l,2}\right) = \mathcal{N}(\mathbf{0}, \tilde{\Phi}^l),$$

Stellenbosch University https://scholar.sun.ac.za

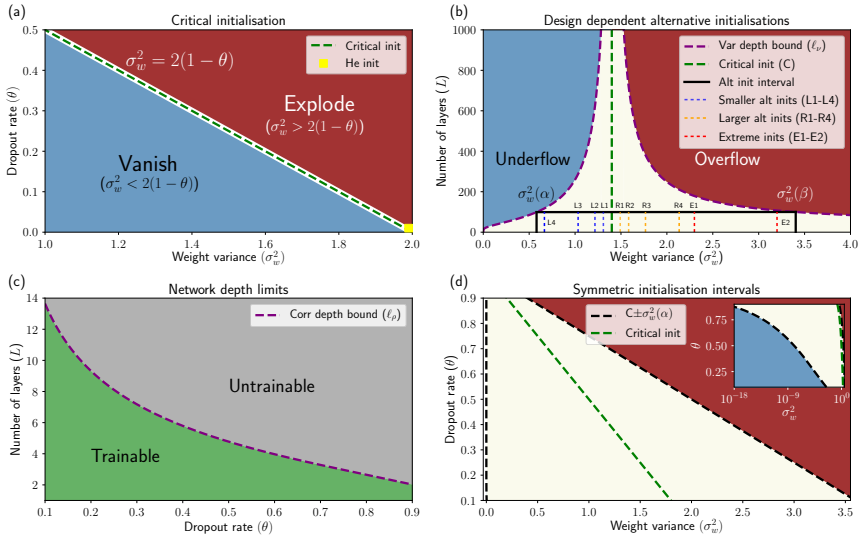


Figure 1: *Choosing network depth and initialisation.* **(a)**: Critical initialisation boundary separating regimes of vanishing or exploding variance signal propagation at large depth. **(b)**: An illustration of the region that allows stable variance (beige) and correlation (black bordered beige) information to propagate through the entire network. (We choose a depth $L = 100$ for ease of visualisation, although this depth is too deep for training with dropout). Alternative non-critical initialisations L1–L4 (blue), C (green), R1–R4 (orange), E1–E2 (red) are sampled from this region. **(c)**: The depth to which networks with dropout are trainable for different dropout rates (dashed purple line). **(d)** Symmetrical interval containing L1–L4 and R1–R4 as a function of the dropout rate: the (beige) region around criticality represents the set of trainable initialisations.

where

$$\tilde{\Phi}^l = \begin{bmatrix} \nu_1^l & \kappa^l \\ \kappa^l & \nu_2^l \end{bmatrix}.$$

The layer-wise evolution of the terms in the covariance matrix $\tilde{\Phi}^l$, are given by

$$\nu_1^l = \frac{\sigma_w^2}{2(1-\theta)}\nu_1^{l-1} + \sigma_b^2 \tag{2}$$

$$\kappa^l = \frac{\sigma_w^2}{2}\kappa^{l-1}\left(g(\rho^{l-1}) + \frac{1}{2}\right) + \sigma_b^2 \tag{3}$$

$$\rho^l = \kappa^l / \sqrt{\nu_1^l \nu_2^l} \tag{4}$$

where

$$g(\rho^{l-1}) = \frac{1}{\pi\rho^{l-1}}\left(\rho^{l-1}\sin^{-1}\left(\rho^{l-1}\right) + \sqrt{1-(\rho^{l-1})^2}\right)$$

with initial variance $\nu_1^0 = \frac{\mathbf{x}_1^0 \cdot \mathbf{x}_1^0}{D^0}$ and covariance $\kappa^0 = \frac{\mathbf{x}_1^0 \cdot \mathbf{x}_2^0}{D^0}$. The above quantities are derived in the large width limit, but in practice tend to hold for finite widths of moderate size (Poole et al., 2016; Schoenholz et al., 2017; Pretorius et al., 2018).

Critical initialisation. A fixed point of the variance recurrence in (2) is given by

$$\{\sigma_w^2, \sigma_b^2\} = \{2(1-\theta), 0\},$$

which ensures that signal propagation variances are preserved during the forward pass of a ReLU network with dropout (Pretorius et al., 2018). These settings of the network parameters is referred to as the *critical initialisation*. Figure 1(a) shows the relationship between the critical initialisation and the dropout rate. Away from criticality, the variance signal tends to vanish or explode. If the dropout rate θ is zero, the initialisation reduces to the popular "He" initialisation for ReLU networks (He et al., 2015).

Trainable depth. Consider the following proposition due to Schoenholz et al. (2017):

> **Proposition 1**: *At initialisation, a necessary condition for training any neural network is that the information from the input layer should be able to reach the output layer.*

Pretorius et al. (2018) analyse the evolution of the input variances and correlations, as given in (2) and (4), to establish when this information propagation requirement is violated. Specifically, let α and β represent the smallest and largest positive values representable on a modern machine. The depth at which numerical instability issues (underflow or overflow) arise from the variances described in (2) for non-critical initialisations

Stellenbosch University https://scholar.sun.ac.za

is bounded by

$$\ell_\nu = \begin{cases} \frac{\ln\left(\frac{\alpha}{\nu 0}\right)}{\ln\left(\frac{\sigma_w^2}{2(1-\theta)}\right)}, & \text{if } \sigma_w^2 < 2(1-\theta) \\ \frac{\ln\left(\frac{\beta}{\nu 0}\right)}{\ln\left(\frac{\sigma_w^2}{2(1-\theta)}\right)}, & \text{if } \sigma_w^2 > 2(1-\theta). \end{cases} \tag{5}$$

An example of these bounds are shown in Figure 1(b) as purple dashed lines. The depth bounds around criticality are finite but large, exceeding typical depths for most modern deep neural networks used in practice. However, even if single input information can propagate to large depths, the correlation between inputs, described in (4), converge to degenerate levels over a much shorter depth horizon (Pretorius et al., 2018). This can limit the network's ability to train, since all discriminatory information is lost during forward propagation. Furthermore, the rate of convergence in correlation is invariant to initialisation, but increases as more dropout is applied. As a result, inputs to a dropout network tend to convey similar information at shallower depths compared to unregularised networks. The bound that characterises convergence in correlation is

$$\ell_\rho = -6/\ln\left[\frac{(1-\theta)}{\pi}\left(\sin^{-1}(\rho^*) + \frac{\pi}{2}\right)\right], \tag{6}$$

where ρ^* denotes the converged correlation and the factor 6 is an ad-hoc factor, which seems to provide a good fit to experimental data, but is as yet unexplained (Schoenholz et al., 2017; Pretorius et al., 2018). Figure 1(c) plots the theoretically predicted trainable depth using (6) for different dropout rates. Note that these depths are much shallower than those derived for variance dynamics.

3 Experimental setup

We conduct a large-scale controlled experiment using networks of trainable depth to compare the effect of initialisation on training speed and generalisation for ReLU networks with dropout. We explore the space around criticality by selecting alternative initialisations whose values theoretically satisfy Proposition 1. Our final aim is to test whether there exists a statistically significant difference, as measured by training speed and generalisation, between the different initialisations. To answer this question, we use a systematic randomised control trial methodology with hypothesis testing.

3.1 Controlled experiments using neural networks: a randomised control trial approach

Inspired by causal discovery in medical research, we consider a hypothesis a priori and conduct a "randomised control trial" (RCT) (Kendall, 2003) using neural networks. In an ordinary randomised control trial a random sample, representative of the full population, is split into two groups. One group receives some form of an intervention, such as a new drug. The other group, referred to as the control group, receives no intervention. The purpose of the two groups is to control for all confounding effects that are unrelated to the intervention of interest. The groups are then monitored by collecting data over time. Once the study has been completed, a test for statistical significance can be applied to ascertain if there exists a difference between the two groups, as measured by a quantitative metric of interest. If a statistical significant difference is detected, the intervention is confirmed as being the cause. In this paper, we aim to test for differences in initialisation of fully-connected ReLU neural networks with dropout.

To begin, consider the following *design space*:

> **Ω-design space:** *We define the neural network design space Ω as the space consisting of different possible combinations of design components used to construct an algorithm for classification using a fully-connected ReLU neural network with dropout. Specifically, the design space is given by the following Cartesian product*
>
> $$\Omega = \mathcal{X} \times \mathcal{D} \times \mathcal{W} \times \mathcal{R} \times \mathcal{B} \times \mathcal{O} \times \mathcal{M} \times \mathcal{L}$$
>
> *where the component sets divide into (1) dataset $\mathcal{X}$, (2) network topology: depth $\mathcal{D}$, width $\mathcal{W}$, (3) dropout rate $\mathcal{R}$, and (4) training procedure: batch size $\mathcal{B}$, optimiser $\mathcal{O}$, momentum $\mathcal{M}$, and learning rate $\mathcal{L}$.*

We adapt the RCT approach for analysing neural network initialisation as follows. First, we randomly generate a collection of different neural network algorithms by sampling from the design space, or "population," of possible neural networks. For example, a 10-layer ReLU network trained on MNIST, where each layer is 256 units wide, with a dropout rate of 0.5, optimised using RMSprop with zero momentum and a learning rate of 5×10^{-4} and batches of size 128, corresponds to the 8-tuple: $(\text{MNIST}, 10, 256, 0.5, 128, \text{RMSprop}, 0, 5 \times 10^{-4})$. Next, we construct identical "groups" by using multiple copies of the sampled designs. Each group in the experiment is then assigned a different initialisation scheme. Finally, we test the following hypothesis related to a given metric:

> **Null hypothesis**: *Given a metric τ, let $\mu_{crit}(\tau)$ denote the group mean associated with*

Stellenbosch University https://scholar.sun.ac.za

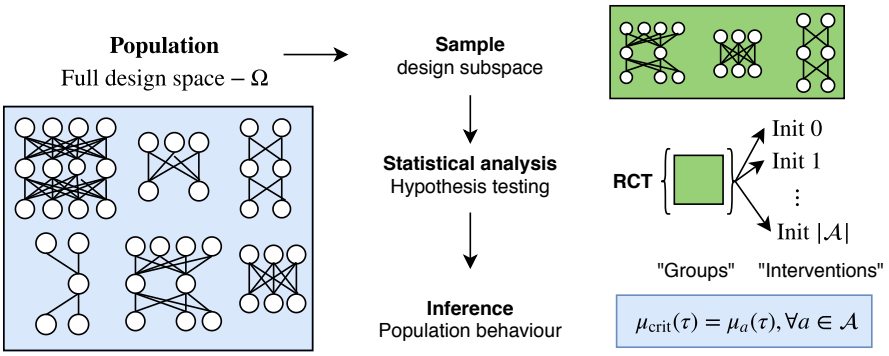


Figure 2: Randomised control trial approach to analysing the effect of initialisation in neural networks.

the critical initialisation and $\mu_a(\tau)$, the mean associated with an alternative initialisation $a \in \mathcal{A} \subset \mathcal{I}$, where $\mathcal{I}$ is the set of all possible initialisations, and $\mathcal{A}$ is our chosen set of alternative initialisations. Then the null hypothesis to be tested is

$$\mathrm{H}_0 : \mu_{\text{crit}}(\tau) = \mu_a(\tau), \forall a \in \mathcal{A}. \qquad (7)$$

We discuss our methodology for selecting alternative initialisations in Section 3.2.

If the null hypothesis is rejected, we have strong evidence to indicate that the performance of the critical initialisation is significantly different from those of alternative initialisations. If H_0 cannot be rejected, the perceived difference is not considered statistically significant. Figure 2 summarises this approach to studying the effect of initialisation in neural networks.

Metrics. Our chosen metrics of interest are training speed and generalisation performance. Specifically, we define these quantities as

- **Training speed** – τ_s: accuracy achieved on the training set at the 100^{th} epoch.
- **Generalisation** – τ_g: highest accuracy achieved on the test set over the course of training.

For example, $\mu_{\text{crit}}(\tau_s)$ denotes the mean training speed associated with the critical initialisation, where a higher mean accuracy at epoch 100 indicates faster training.

Sampling algorithm designs. For our experiment groups, we sample 1120 different designs. These designs are drawn randomly from Ω, which we construct by forming the Cartesian product of the following discrete sets for dataset, depth, width, dropout rate, batch size, optimiser, momentum and learning rate:

$$\begin{aligned}
\mathcal{X} &= \{\text{MNIST}, \text{FashionMNIST}, \text{CIFAR-10}, \text{CIFAR-100}\} \\
\mathcal{D} &= \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 20\} \\
\mathcal{W} &= \{400, 600, 800\} \\
\mathcal{R} &= \{0,\ 0.1,\ 0.3,\ 0.5\} \\
\mathcal{B} &= \{32, 64, 128, 256\} \\
\mathcal{O} &= \{\text{SGD}, \text{Adam}, \text{RMSprop}\} \\
\mathcal{M} &= \{0, 0.5, 0.9\} \\
\mathcal{L} &= \{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}
\end{aligned}$$

For a given dropout rate, we limit the sampled depths in $\mathcal{D}$ to only the settings that would allow useful correlation information to reach the output layer, i.e. $d \leq \ell_\rho, \forall d \in \mathcal{D}$. We also include network depths of 15 and 20 when no dropout is being applied. We sample 70 designs for each dropout rate and dataset combination for a large enough diversity in network architecture and optimisation. To ensure a balanced set of network designs, we simply duplicate each group of designs for every dropout rate in $\mathcal{R}$, as well as for each dataset. A full description of this process is presented in Appendix A. Finally, each network is trained for 500 epochs on MNIST (LeCun et al., 1998), FashionMNIST (Xiao et al., 2017), CIFAR-10 and CIFAR-100 (Krizhevsky and Hinton, 2009), using the full training set for each.

3.2 A network design dependent set of alternative initialisations

We ensure that our networks preserve short range correlation information by limiting their depth. To develop a principled approach towards exploring the initialisation space around criticality, we now ask: for a fixed depth, what is the range of initialisations around criticality that will remain numerically stable until the output layer? In other words, for which σ_w^2 in (5) is $\ell_\nu \geq \ell_\rho$. We can find these bounds for alternative initialisations by solving for σ_w^2 in (5), which gives

Stellenbosch University https://scholar.sun.ac.za

$$\sigma_w^2(\alpha) = \inf\left\{\sigma_w^2 \in \mathbb{R}_{>0} \middle| \ell_\nu \geq \ell_\rho, \sigma_w^2 < 2(1-\theta)\right\}$$
$$= 2(1-\theta)\left(\frac{\alpha}{\nu^0}\right)^{1/\ell_\rho} \quad \text{[lower bound]} \tag{8}$$
$$\sigma_w^2(\beta) = \sup\left\{\sigma_w^2 \in \mathbb{R}_{>0} \middle| \ell_\nu \geq \ell_\rho, \sigma_w^2 > 2(1-\theta)\right\}$$
$$= 2(1-\theta)\left(\frac{\beta}{\nu^0}\right)^{1/\ell_\rho} \quad \text{[upper bound]} \tag{9}$$

In our experiments, we use 32-bit floating point precision, such that $\alpha = 1.1754944 \times 10^{-38}$ in (8) and $\beta = 3.4028235 \times 10^{38}$ in (9). An example interval for possible alternative initialisations bounded by $\sigma_w^2(\alpha)$ and $\sigma_w^2(\beta)$ is shown in Figure 1(b). Note that the interval is not symmetric. This is because although the signal vanishes and explodes at the same rate, the critical initialisation is typically much closer to α than to β. This causes the interval to be wider to the right. To cover this entire space around criticality would be computationally infeasible. Therefore, we focus on sampling alternative initialisations around criticality as a function of the dropout rate.

Specifically, we first sample a core set of initialisations within the interval $\mathrm{C} \pm \sigma_w^2(\alpha)$ centred around the critical initialisation (C), with logarithmic spacing between samples (see Appendices B and C for more detail). This symmetric interval is illustrated by the dashed black lines in Figure 1(d) for different dropout rates. Note that the interval becomes narrower for larger dropout rates. The inset in Figure 1(d) plots the left side of the interval close to zero on a log-scale. The core set of alternative initialisations for the fixed depth in Figure 1(b) are shown as blue dashed lines (below criticality, marked L1-L4) and orange dashed lines (above criticality, marked R1-R4). Finally, we explore further to the right by sampling halfway, as well as close to the end of the interval, between criticality and $\sigma_w^2(\beta)$. These more extreme initialisations are depicted in red (marked E1 and E2).

3.3 Statistical comparison methodology

The null hypothesis H_0 can be tested using an *omnibus test*, which is specifically designed for multiple comparisons (Demšar, 2006). If the null hypothesis is rejected in this setting, there is evidence to suggest that at least one of the competing initialisations is significantly different from the rest. Specifically, we use the Iman-Davenport extension (Iman and Davenport, 1980) of the non-parametric Friedman rank test (Friedman, 1937) as recommended by Demšar (2006) and García et al. (2010). We describe this test below in the context of comparing different initialisations.

- **Friedman** (Friedman, 1937): For a given metric τ and a set of competing initialisations $\mathcal{I}$, the Friedman test first ranks initialisations $i \in \mathcal{I}$ in terms of their mean performances $\mu_i(\tau)$ and then computes a test statistic using these ranks. An average rank is assigned to tied initialisations. In more detail, let r_{di} denote the rank for a specific design d (sampled from Ω) using initialisation i. We denote the mean rank over the set of all sampled designs $\Delta \subset \Omega$, as $\bar{R}_i = \frac{1}{|\Delta|}\sum_{d=1}^{|\Delta|} r_{di}$. The Friedman test statistic under the null hypothesis of no difference is
$$\chi_F^2 = \frac{12|\Delta|}{|\mathcal{I}|(|\mathcal{I}|+1)}\left(\sum_{i=1}^{|\mathcal{I}|}\bar{R}_i^2 - \frac{|\mathcal{I}|(|\mathcal{I}|+1)^2}{4}\right),$$
and is approximately χ^2 distributed with $|\mathcal{I}|-1$ degrees of freedom.
- **Iman-Davenport** (Iman and Davenport, 1980): It has been shown that the Friedman test can be a poor approximation to the χ^2 distribution. Therefore, the Iman-Davenport test modifies the Friedman test as follows
$$F_{ID} = \frac{(|\Delta|-1)\chi_F^2}{|\Delta|(|\mathcal{I}|-1)-\chi_F^2},$$
to more accurately approximate an F distribution with $(|\mathcal{I}|-1)$ and $(|\mathcal{I}|-1)(|\Delta|-1)$ degrees of freedom.

If we reject H_0, we may next ask whether there exists specific differences between the critical and the alternative initialisations. For this purpose we perform multiple pairwise tests.

It is important to note, however, that when conducting multiple pairwise comparisons with popular two-sample tests, a significant difference might be detected simply by chance. To illustrate this, consider the probability of rejecting the null hypothesis when it is in fact true. This is known as a type I error. The null hypothesis is usually rejected if the probability of a type I error—the p-value—is less than some specified *significance level*, typically set at 5%. However, it is insufficient to separately control for type I errors for each individual pairwise comparison. In our case, pairwise comparisons between the critical and the alternative initialisations (L1–L4, R1–R4, E1, E2) result in a total of 10 comparisons. At a significance level of 5%, a satisfactory probability of not making a type I error in a *single* comparison is $\gamma = 1 - p(\text{reject } \mathrm{H}_0|\mathrm{H}_0 \text{ is true}) = 95\%$. However, the probability of not making a type I error *across all comparisons* is actually $\gamma^{10} \approx 60\%$, which is much lower than what was previously considered acceptable. Therefore, we guard against type I errors in multiple

Stellenbosch University https://scholar.sun.ac.za

tests by using *post-hoc* tests that aim to adjust the significance level to control the *family-wise* error—the probability that at least one type I error is made among multiple tests (García et al., 2010; Santafe et al., 2015). The specific post-hoc test we use is the Finner test (Finner, 1993) as recommended by Garcia and Herrera (2008) and García et al. (2010). The specifics of this test are given below.

- **Finner** (Finner, 1993): Let p_i, $i = 1, ..., i^*, ..., |\mathcal{I}|$, denote ordered p-values obtained from multiple pairwise comparisons corresponding to the null hypotheses of no mean difference $\mathrm{H}_{01}, ..., \mathrm{H}_{0i^*}, ..., \mathrm{H}_{0|\mathcal{I}|}$. Using the Finner test, we reject $\mathrm{H}_{01}, ..., \mathrm{H}_{0i^*}$, where

$$i^* = \min\left\{ i | p_i > 1 - \gamma^{i/(|\mathcal{I}|-1)} \right\}.$$

3.4 Summary of experimental setup

We aim to test for differences in initialisation by conducting a large scale randomised control trial experiment using neural networks. We begin by sampling 70 neural network algorithm designs from the design space Ω for each dropout rate and dataset combination, for a total of 1120 designs. To form groups in our experiment, we make 11 identical copies of the 1120 designs. Note that this is a core aspect of our approach. We ensure *within-group variation* by sampling *different designs*, but then duplicate this collection of designs to form identical groups, one for each "intervention", i.e. initialisation. For each group, we assign a different initialisation—either critical initialisation (C), or one of the 10 alternative initialisations (L1–L4, R1–R4, E1–E2). All designs are then trained, resulting in a total of $70 \times 4 \times 4 \times 11 = 12320$ trained neural networks. Using these results, we test our hypothesis—that no difference exists between the various initialisations in terms of training speed and generalisation—using omnibus and post-hoc statistical tests.

To provide an analogy in the context of drug testing: our approach is akin to selecting a large random sample of human participants, duplicating (cloning) them to form identical groups, and then administering a different drug to each group. To have exact copies of a representative sample to test on is an ideal case for an experimenter, since (1) within-group variation controls for confounding effects, and (2) having identical groups ensures that if differences between groups are detected, it can only be as a result of the drug. Here we are fortunate to be dealing with software entities and not human beings, which allows us the luxury of this ideal setup (see Appendix E for a further discussion on the validity of the RCT approach).

4 Results

A visualisation of our findings is presented in Figure 3. For each initialisation, we plot densities summarising the results for (a) training speed and (b) generalisation. Visually our analysis seems to indicate that the average effect on training speed and generalisation for the critical initialisation is quite similar to the average effect of alternative initialisations, except at the extremes. To make this conclusion more concrete, we conduct a statistical analysis of the results.

Statistical analysis.[1] Using the Iman-Davenport omnibus test, we reject the null hypothesis of no difference between the different initialisations for both training speed and generalisation, with a p-value equal to 2.2×10^{-16} (practically zero). This is somewhat unsurprising, since there are clear differences between the initialisations closer to criticality and those at the extremes (E1 and E2). Therefore, given that we have rejected H_0, we also conduct post-hoc tests.

Table 1 provides pairwise comparisons between the critical initialisation and the alternatives. For mean training speed, we find that only the initialisations at the extremes, i.e. close to $\sigma_w^2(\alpha)$ and $\sigma_w^2(\beta)$, give significantly different results. These include initialising very close to zero (L4) and very large initialisations (E1 and E2). For the initialisations around criticality the differences are not statistically significant. For generalisation, it seems that the alternative initialisations are more sensitive to deviations from criticality (only R1 and L3 indicate no statistically significant difference). However, given the large scale of our study we are able to detect very fine differences. Therefore, even when differences are significant, it is important to consider the sizes of their effects.

Effect sizes. The purpose of computing effect sizes is to gauge whether statistically significant differences in effects are actually meaningful as measured by their magnitude. For a metric τ, we define the effect size for an alternative initialisation $a \in \mathcal{A}$, as $d_a(\tau) = [\mu_a(\tau) - \mu_{\text{crit}}(\tau)]/\text{sd}_{\text{crit}}(\tau)$, where $\text{sd}_{\text{crit}}(\tau)$ is the standard deviation of τ for the critical initialisation. This definition of effect size for a given quantity is often referred to as Cohen's d (Cohen, 1988). In our context, a value of $d = 1$ can be interpreted as a difference in effects equal to one standard deviation away from the mean of criticality. Effect sizes are typically considered to be large, i.e. meaningful, for $d \geq 0.8$. The effect sizes for all the alternative initialisations are given in Table 1, where the direction of an effect is indicated

[1] Although the results appear to be multimodal, our non-parametric tests are based on ranking and therefore do not make assumptions regarding the underlying distribution. Our tests are therefore still appropriate.

Stellenbosch University https://scholar.sun.ac.za

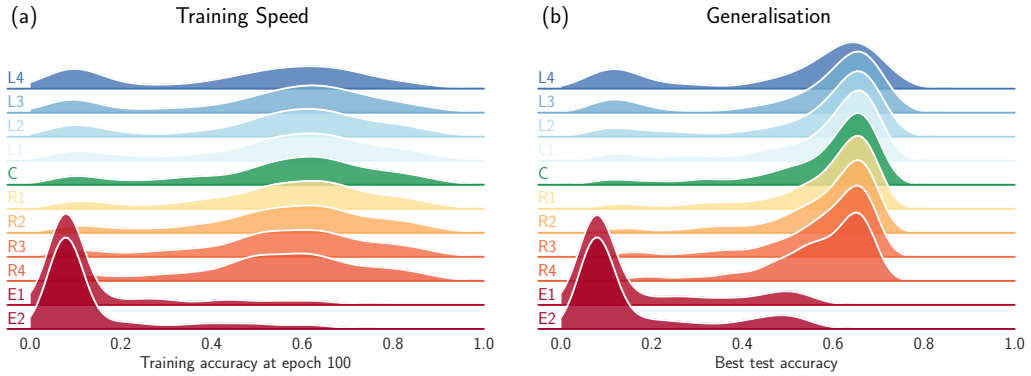


Figure 3: *Density fits for the different initialisations.* **(a)**: training speed. **(b)**: generalisation.

Table 1: Post-hoc tests for training speed and generalisation. We use the symbols * to indicate a significant p-value (less than 5%) and † for a large effect size.

TRAINING SPEED – $H_0 : \mu_{\text{CRIT}}(\tau_s) = \mu_a(\tau_s), a \in \{L_i, R_i, E_1, E_2\}_{i=1}^4$

	L4	L3	L2	L1	R1	R2	R3	R4	E1	E2
ADJUSTED p-VALUE	$^*\mathbf{0}$	0.1338	0.857	0.4003	0.6479	0.0677	$^*\mathbf{3.95 \times 10^{-6}}$	$^*\mathbf{3.33 \times 10^{-15}}$	$^*\mathbf{2.4 \times 10^{-9}}$	$^*\mathbf{2.4 \times 10^{-9}}$
EFFECT SIZE	−0.2512	−0.0648	−0.0287	−0.0069	0.0094	0.0202	0.0147	−0.0048	$^\dagger\mathbf{-1.1055}$	$^\dagger\mathbf{-1.1019}$

GENERALISATION – $H_0 : \mu_{\text{CRIT}}(\tau_g) = \mu_a(\tau_g), a \in \{L_i, R_i, E_1, E_2\}_{i=1}^4$

	L4	L3	L2	L1	R1	R2	R3	R4	E1	E2
ADJUSTED p-VALUE	$^*\mathbf{0}$	0.0545	$^*\mathbf{0.0033}$	$^*\mathbf{0.0418}$	0.1226	$^*\mathbf{9.23 \times 10^{-6}}$	$^*\mathbf{4.44 \times 10^{-16}}$	$^*\mathbf{0}$	$^*\mathbf{0}$	$^*\mathbf{0}$
EFFECT SIZE	−0.2389	−0.0716	−0.0412	−0.0103	−0.0016	0.0102	0.0049	−0.0145	$^\dagger\mathbf{-1.1708}$	$^\dagger\mathbf{-1.1787}$

by its sign (negative indicating a worse performance when compared to criticality). Effect sizes larger than 0.8 in absolute value are marked with the † symbol. As suggested by the plots in Figure 3, the majority of initialisations around criticality with statistically significant differences in generalisation, as shown in Table 1, have negligible effects sizes. The only meaningful effects are again at the extremes. Finally, we note that the above findings also hold when just considering standard ReLU networks without dropout (shown in Appendix D).

5 Conclusion

At large depth, critical initialisation for neural networks is often considered crucial for success in training and generalisation. However, recent work has shown that dropout, a popular regularisation strategy, limits the depth to which networks can be trained. Given this depth limit, we explore the question of whether initialising at criticality still matters. Or whether it is possible that alternative, non-critical initialisations (less suited for stable signal propagation at extreme depth) provide any previously undiscovered benefits over critical initialisation. We conducted a large-scale controlled experiment by training over 12000 neural networks. A systematic statistical analysis of training speed and generalisation performance showed that, for a wide range of alternative initialisations around criticality, there is no statistically significant difference between these initialisations and the critical initialisation. Our analysis provides strong evidence that, for moderately deep feedforward ReLU networks (as well as those whose depth is constrained by dropout), there is little to be gained by searching for alternative initialisation schemes.

We emphasize the value of the methodology presented in this paper. Initialisation aside, the methodology can be followed in a generic way to rigorously test the effects of any design component of interest associated with a particular machine learning algorithm. Since statistical rigour is often lacking in empirical machine learning research, we hope that this approach might serve as a useful template for more rigorous future investigations.

Stellenbosch University https://scholar.sun.ac.za

References

N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

S. Wang and C. Manning, "Fast dropout training," in *Proceedings of the 30th International Conference on Machine Learning*, 2013, pp. 118–126.

S. Wager, S. Wang, and P. S. Liang, "Dropout training as adaptive regularization," in *Advances in Neural Information Processing Systems*, 2013, pp. 351–359.

P. Baldi and P. J. Sadowski, "Understanding dropout," in *Advances in Neural Information Processing Systems*, 2013, pp. 2814–2822.

L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on Machine Learning*, 2013, pp. 1058–1066.

Y. Gal, J. Hron, and A. Kendall, "Concrete dropout," in *Advances in Neural Information Processing Systems*, 2017, pp. 3581–3590.

A. N. Gomez, I. Zhang, K. Swersky, Y. Gal, and G. E. Hinton, "Targeted dropout," 2018.

G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Dropblock: A regularization method for convolutional networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 727–10 737.

D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Advances in Neural Information Processing Systems*, 2015, pp. 2575–2583.

Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 2016, pp. 1050–1059.

D. Molchanov, A. Ashukha, and D. Vetrov, "Variational dropout sparsifies deep neural networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2498–2507.

S. S. Schoenholz, J. Gilmer, S. Ganguli, and J. Sohl-Dickstein, "Deep information propagation," *Proceedings of the International Conference on Learning Representations*, 2017.

J. Pennington, S. Schoenholz, and S. Ganguli, "Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice," in *Advances in Neural Information Processing Systems*, 2017, pp. 4788–4798.

L. Xiao, Y. Bahri, J. Sohl-Dickstein, S. S. Schoenholz, and J. Pennington, "Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks," *Proceedings of the International Conference on Machine Learning*, 2018.

M. Chen, J. Pennington, and S. S. Schoenholz, "Dynamical isometry and a mean field theory of RNNs: Gating enables signal propagation in recurrent neural networks," *Proceedings of the International Conference on Machine Learning*, 2018.

A. Pretorius, E. Van Biljon, S. Kroon, and H. Kamper, "Critical initialisation for deep signal propagation in noisy rectifier neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 5722–5731.

D. Hendrycks and K. Gimpel, "Adjusting for dropout variance in batch normalization and weight initialization," *arXiv preprint arXiv:1607.02488*, 2016.

A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *Proceedings of the International Conference on Learning Representations*, 2014.

K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli, "Exponential expressivity in deep neural networks through transient chaos," in *Advances in Neural Information Processing Systems*, 2016, pp. 3360–3368.

J. Kendall, "Designing a research project: randomised controlled trials and their principles," *Emergency Medicine Journal*, vol. 20, no. 2, pp. 164–168, 2003.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

Stellenbosch University https://scholar.sun.ac.za

R. L. Iman and J. M. Davenport, "Approximations of the critical region of the fbietkan statistic," *Communications in Statistics-Theory and Methods*, vol. 9, no. 6, pp. 571–595, 1980.

M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the american statistical association*, vol. 32, no. 200, pp. 675–701, 1937.

S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Information Sciences*, vol. 180, no. 10, pp. 2044–2064, 2010.

G. Santafe, I. Inza, and J. A. Lozano, "Dealing with the evaluation of supervised classification algorithms," *Artificial Intelligence Review*, vol. 44, no. 4, pp. 467–508, 2015.

H. Finner, "On a monotonicity problem in step-down multiple test procedures," *Journal of the American Statistical Association*, vol. 88, no. 423, pp. 920–923, 1993.

S. Garcia and F. Herrera, "An extension on"statistical comparisons of classifiers over multiple data sets"for all pairwise comparisons," *Journal of Machine Learning Research*, vol. 9, no. Dec, pp. 2677–2694, 2008.

J. Cohen, *Statistical power analysis for the behavioral sciences*. 2d ed. Hillsdale, N.J.: Lawrence Erlbaum., 1988.

X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

Stellenbosch University https://scholar.sun.ac.za

Appendix

A Pseudo random design construction

A.1 Initial / stage-one designs

First we construct incomplete neural network designs by taking the random Cartesian product of the form:

$$\Omega_* = \mathcal{W} \times \mathcal{B} \times \mathcal{O} \times \mathcal{M} \times \mathcal{L}. \tag{10}$$

To ensure balanced designs are created (in the sense that the design space contains an equal number of each item in any of the above sets), we do not sample from the sets with a uniform random probability of selecting each element, but rather concatenate random permutations of each set and pair configurations across this. This process is best illustrated with a small example:

Suppose we only have two hyper-parameter choices: the width of the hidden layers and the learning rate. We would then want to generate pairs of layer-width-learning-rate samples. Our method is to:

1. concatenate random permutations of each set:
   $\widehat{\mathcal{W}} = [\text{permute}(\mathcal{W}), \ldots, \text{permute}(\mathcal{W})] = [600, 400, 800, 800, 600, 400, 600, 400]$ (for example)
   $\widehat{\mathcal{L}} = [\text{permute}(\mathcal{L}), \ldots, \text{permute}(\mathcal{L})] = [10^{-4}, 10^{-3}, 10^{-5}, 10^{-6}, 10^{-3}, 10^{-6}, 10^{-5}, 10^{-4}]$ (for example)

2. sequentially pair these concatenated sets:
   $\Omega_*^{(i)} = \left(\widehat{\mathcal{W}}^{(i)}, \widehat{\mathcal{L}}^{(i)}\right)$
   $\therefore \Omega_*^{(0)} = \left(600, 10^{-4}\right); \Omega_i^{(1)} = \left(400, 10^{-3}\right)$; etc

We then duplicate these combinations for each dropout rate we wish to test. Subsequently, we generate the set of viable correlation information preserving depths based on the dropout rate that is present in each configuration. We use the same setup as above to pair viable depths with incomplete combinations to form complete combinations.

Note that Adam does not support momentum. Thus, when Adam and momentum values were paired, the momentum parameter was ignored when creating the network.

A.2 Complete / stage-two designs

A process identical to the above is followed to match incomplete designs Ω_* with viable network depths based on the dropout rate of the group. We construct depth sets $\mathcal{D}_\theta \subset \mathcal{D}$ such that $\mathcal{D}_\theta$ only contains depths to which correlation information can propagate for the given dropout rate θ –see Figure 4 for a graphical representation of this and note that, since we are working with networks with discrete numbers of layers, we follow the discretised boundary. We then concatenate permutations of this set and use this to complete the designs and form our final designs Ω. Let us illustrate this by continuing the above example for $\theta = 0.7$:

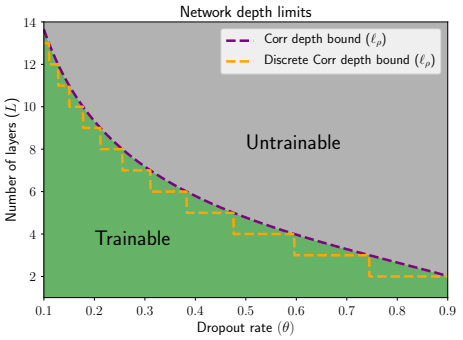


Figure 4: *Discrete trainable depth boundary.* The depth to which networks with dropout are trainable for different dropout rates (dashed purple line). Discrete depths for each dropout rate is shown by the dashed orange line. When creating networks in practice, the discrete bound should be considered.

1. construct $\mathcal{D}_{0.7} = \{2, 3, 4, 5, 6, 7\}$ using (6) (we omit networks with 0 and 1 hidden layers to ensure some measure of expressiblity)

2. concatenate random permutations of $\mathcal{D}_{0.7}$:
   $\widehat{\mathcal{D}}_{0.7} = [\text{permute}(\mathcal{D}_{0.7}), \ldots, \text{permute}(\mathcal{D}_{0.7})] = [6, 3, 7, 2, 5, 4, 2, 3, 6, 5, 4, 7]$ (for example)

3. sequentially pair this concatenated set with the incomplete designs:
   $\Omega^{(i)} = \left(\Omega_*^{(i)}, \widehat{\mathcal{D}}_{0.7}^{(i)}, 0.7\right)$
   $\therefore \quad \Omega^{(0)} = \left(600, 10^{-4}, 6, 0.7\right); \Omega^{(1)} = \left(400, 10^{-3}, 3, 0.7\right)$; etc

These four design sets, one for each value in $\mathcal{R}$, are then duplicated 44 times (once for each combination of initialisation candidate in $\mathcal{A}$ and dataset in $\mathcal{X}$).

B Method for generating network design dependent initialisations

Inputs:

- θ: dropout rate
- L: depth of the network
- S: the number of candidates on either side of critical, within the core group, to be generated

Stellenbosch University https://scholar.sun.ac.za

- E: the number of "extreme" candidates (those far greater than critical) to be generated
- β: the largest positive value that can be represented given the floating point precision of the current computer
- α: the smallest positive value that can be represented given the floating point precision of the current computer

Steps:

1. calculate $\sigma^2_{\text{critical}} = 2(1-\theta)$
2. calculate $\sigma^2_w(\beta)$ using (10)
3. generate the set of "extreme" samples, $\boldsymbol{\sigma^2_{\text{extreme}}}$:
   (a) select the first "extreme" candidate such that it is within the depth boundary: $\sigma^2_{\text{extreme},E} = 0.9\sigma^2_w(\beta)$
   (b) recursively calculate the subsequent "extreme" candidates such that they are logarithmically spaced: $\sigma^2_{\text{extreme},e} = \frac{1}{2}\sigma^2_{\text{extreme},(e+1)}$ for $e \in \{1, 2, ..., E-1\}$
4. calculate $\sigma^2_w(\alpha)$ using (9)
5. generate the set of logarithmically spaced samples less than critical, $\boldsymbol{\sigma^2_{\text{left}}}$: $\sigma^2_{\text{left},s} = \sigma^2_{\text{critical}} - \frac{0.9}{2^{s-1}}(\sigma^2_{\text{critical}} - \sigma^2_w(\alpha))$ for $s \in \{1, 2, ..., S\}$
6. generate the set of samples just greater than critical, $\boldsymbol{\sigma^2_{\text{right}}}$, by reflecting $\boldsymbol{\sigma^2_{\text{left}}}$ about the critical initialisation: $\sigma^2_{right,s} = \sigma^2_{\text{critical}} - (\sigma^2_{left,s} - \sigma^2_{\text{critical}}) = \sigma^2_{\text{critical}} + \frac{0.9}{2^{s-1}}(\sigma^2_{\text{critical}} - \sigma^2_w(\alpha))$ for $s \in \{1, 2, ..., S\}$

The set of candidate initialisations is then $\left\{\boldsymbol{\sigma^2_{\text{left}}}, \sigma^2_{\text{critical}}, \boldsymbol{\sigma^2_{\text{right}}}, \boldsymbol{\sigma^2_{\text{extreme}}}\right\}$.

C Design and corresponding initialisation examples

Table 2 shows 12 sampled designs and their corresponding initialisations. These samples are representative of our full set of design samples and give a good idea of typical network parameters. While there appears to be no difference between core initialisation values across samples with the same dropout rate, this is actually not the case. Changes are simply typically too small to be seen with only 3 decimal places. This is due to the rate of change of $\sigma^2_w(\alpha)$ being very low for networks of shallow to moderate depth (roughly 20 hidden layers or less).

D Additional results

In this section we provide additional statistical analyses per dropout rate as well as with zero dropout. These results are given in Tables 3, 4, 5 and 6.

E On the validity of the RCT approach

We performed two auxiliary studies to ensure the effectiveness of the RCT setup.

Firstly, we wanted to ensure the methodology was set up correctly and could identify known performance differences. To achieve this, we created a smaller scale scenario very similar to the study described in the main text but instead using initialisation as "intervention", we use the activation function. Networks with non-linear activation functions are more expressive and should be able to outperform their linear counterparts. Furthermore, the ReLU activation function does not suffer from saturation or vanishing gradients and typically outperforms the sigmoid when using random Gaussian initialisation. These are well established results, frequently demonstrated in the literature. Therefore, we decided to construct an RCT where the interventions are the following activations: linear, sigmoid, and ReLU. Each network was initialised critically and trained on MNIST for 1955 iterations using a batch size of 128. This RCT consisted of 1761 trained networks.

The results of the above experiment are exactly as expected and are given in Figure 5. ReLU networks performed best overall. The performance of linear networks were capped significantly below that of ReLU networks. Finally, the sigmoidal networks were able to perform better than linear networks and as well as ReLU networks in the best case. However, the distribution over performance for the sigmoid exhibits a long tail towards low accuracy due to vanishing gradient issues, causing network training to stall. Furthermore, pairwise comparisons with post-hoc tests between ReLU and the other activations yield p-values that are practically zero, indicating significantly different mean performances with meaningful effects sizes (-2.65 for linear and -8.42 for sigmoid).

After confirming the validity of our RCT setup, we aimed to control for more sophisticated training procedures, such as learning rate decay. Learning rate decay is expected to have a complex temporal interaction with other design components during training and poses a challenge as a potential confounder for the RCT to control. We theorise that adding any mechanism that generally improves performance, such as learning rate decay, should have an average effect (taken over sampled designs) that is roughly equal across groups.

To test this, we construct two RCTs nearly identical

Stellenbosch University https://scholar.sun.ac.za

Table 2: 12 example sets of sampled designs and their corresponding initialisations.

– Example designs –

design index	dataset	depth	width	rate	batch size	optimiser	momentum	learning rate
0	CIFAR-10	12	800	0.0	256	Adam	0.0	10^{-5}
1	FashionMNIST	15	800	0.0	64	SGD	0.5	10^{-6}
2	MNIST	4	400	0.0	256	RMSprop	0.0	10^{-5}
3	CIFAR-10	2	400	0.5	256	RMSprop	0.0	10^{-5}
4	CIFAR-100	3	800	0.5	64	SGD	0.5	10^{-6}
5	FashionMNIST	4	800	0.5	256	Adam	0.0	10^{-5}
6	CIFAR-10	7	600	0.3	256	Adam	0.0	10^{-3}
7	CIFAR-100	4	600	0.3	64	SGD	0.5	10^{-6}
8	FashionMNIST	5	400	0.3	256	RMSprop	0.0	10^{-5}
9	CIFAR-10	3	600	0.1	32	SGD	0.5	10^{-4}
10	MNIST	8	600	0.1	32	SGD	0.5	10^{-4}
11	CIFAR-10	4	600	0.1	128	Adam	0.0	10^{-3}

– Initialisations –

design index	L4	L3	L2	L1	C	R1	R2	R3	R4	E1	E2
0	0.201	1.101	1.550	1.775	2.000	2.225	2.450	2.899	3.799	1.464×10^{3}	2.926×10^{3}
1	0.205	1.103	1.551	1.776	2.000	2.224	2.449	2.897	3.795	3.346×10^{2}	6.671×10^{2}
2	0.200	1.100	1.550	1.775	2.000	2.225	2.450	2.900	3.800	3.865×10^{9}	7.731×10^{9}
3	0.100	0.550	0.775	0.887	1.000	1.113	1.225	1.450	1.900	8.301×10^{18}	1.660×10^{19}
4	0.100	0.550	0.775	0.888	1.000	1.112	1.225	1.450	1.900	3.142×10^{12}	6.283×10^{12}
5	0.100	0.550	0.775	0.888	1.000	1.112	1.225	1.450	1.900	1.933×10^{9}	3.865×10^{9}
6	0.140	0.770	1.085	1.243	1.400	1.557	1.715	2.030	2.660	2.013×10^{5}	4.026×10^{5}
7	0.140	0.770	1.085	1.243	1.400	1.557	1.715	2.030	2.660	2.706×10^{9}	5.412×10^{9}
8	0.140	0.770	1.085	1.243	1.400	1.557	1.7154	2.030	2.660	3.204×10^{7}	6.408×10^{7}
9	0.180	0.990	1.395	1.598	1.800	2.002	2.205	2.610	3.4204	5.655×10^{12}	1.131×10^{13}
10	0.180	0.990	1.395	1.598	1.800	2.002	2.205	2.610	3.420	5.309×10^{4}	1.062×10^{5}
11	0.180	0.990	1.395	1.598	1.800	2.002	2.205	2.610	3.420	3.479×10^{9}	6.958×10^{9}

Table 3: **No dropout** $-\ \theta = 0$: Post-hoc tests for training speed and generalisation for no dropout ($\theta = 0$). The symbols * indicate a significant p-value and † a large effect size.

Training speed – $H_0 : \mu_{\text{CRIT}}(\tau_s) = \mu_a(\tau_s), a \in \{L_i, R_i, E_1, E_2\}_{i=1}^4$

	L4	L3	L2	L1	R1	R2	R3	R4	E1	E2
Adjusted p-value	$^*\mathbf{0}$	$\mathbf{1.99 \times 10^{-11}}$	$\mathbf{3.07 \times 10^{-5}}$	0.0864	0.0529	$\mathbf{9.68 \times 10^{-5}}$	$\mathbf{8.89 \times 10^{-6}}$	$\mathbf{4.71 \times 10^{-7}}$	$^*\mathbf{0}$	$^*\mathbf{0}$
Effect size	−0.6024	−0.1854	−0.1124	−0.0434	0.0382	0.0716	0.0774	0.0602	$^\dagger\mathbf{-1.1085}$	$^\dagger\mathbf{-1.1009}$

Generalisation – $H_0 : \mu_{\text{CRIT}}(\tau_g) = \mu_a(\tau_g), a \in \{L_i, R_i, E_1, E_2\}_{i=1}^4$

	L4	L3	L2	L1	R1	R2	R3	R4	E1	E2
Adjusted p-value	$^*\mathbf{0}$	$^*\mathbf{0.0033}$	0.8549	0.7927	0.1846	$^*\mathbf{0.0018}$	$^*\mathbf{7.20 \times 10^{-6}}$	$^*\mathbf{3.49 \times 10^{-14}}$	$^*\mathbf{0}$	$\mathbf{0}$
Effect size	−0.5224	−0.1592	−0.0975	−0.0325	−0.0027	0.0077	−0.0032	−0.0374	$^\dagger\mathbf{-1.0008}$	$^\dagger\mathbf{-1.0191}$

Table 4: **Dropout** $-\ \theta = 0.1$: Post-hoc tests for training speed and generalisation for dropout with rate $\theta = 0.5$. The symbols * indicate a significant p-value and † a large effect size.

Training speed – $H_0 : \mu_{\text{CRIT}}(\tau_s) = \mu_a(\tau_s), a \in \{L_i, R_i, E_1, E_2\}_{i=1}^4$

	L4	L3	L2	L1	R1	R2	R3	R4	E1	E2
Adjusted p-value	$^*\mathbf{0}$	$^*\mathbf{0.0203}$	0.9219	0.6552	0.9395	0.2258	$^*\mathbf{0.0021}$	$^*\mathbf{2.59 \times 10^{-8}}$	$^*\mathbf{0}$	$^*\mathbf{0}$
Effect size	−0.2578	−0.0723	−0.0189	0.0015	0.0311	0.0261	0.0202	−0.0009	$^\dagger\mathbf{-1.1328}$	$^\dagger\mathbf{-1.1209}$

Generalisation – $H_0 : \mu_{\text{CRIT}}(\tau_g) = \mu_a(\tau_g), a \in \{L_i, R_i, E_1, E_2\}_{i=1}^4$

	L4	L3	L2	L1	R1	R2	R3	R4	E1	E2
Adjusted p-value	$^*\mathbf{0}$	0.4475	0.8743	0.7033	0.7874	0.5100	$^*\mathbf{0.0193}$	$^*\mathbf{1.41 \times 10^{-6}}$	$^*\mathbf{0}$	$^*\mathbf{0}$
Effect size	−0.3091	−0.0976	−0.0650	−0.0237	0.0139	0.0236	0.0221	0.0167	$^\dagger\mathbf{-1.1503}$	$^\dagger\mathbf{-1.1572}$

Stellenbosch University https://scholar.sun.ac.za

Table 5: **Dropout** $-\ \theta = 0.3$: Post-hoc tests for training speed and generalisation for dropout with rate $\theta = 0$. The symbols * indicate a significant p-value and † a large effect size.

TRAINING SPEED $-$ $H_0 : \mu_{CRIT}(\tau_s) = \mu_a(\tau_s), a \in \{L_i, R_i, E_1, E_2\}_{i=1}^4$

	L4	L3	L2	L1	R1	R2	R3	R4	E1	E2
ADJUSTED p-VALUE	$^*\mathbf{0.0051}$	0.0727	0.0727	0.2097	0.1885	$^*\mathbf{0.0090}$	$^*\mathbf{1.49 \times 10^{-6}}$	$^*\mathbf{4.88 \times 10^{-14}}$	$^*\mathbf{0}$	$^*\mathbf{0}$
EFFECT SIZE	-0.1348	-0.0281	-0.0020	-0.0009	-0.0066	-0.0047	-0.0124	-0.0336	$^\dagger\mathbf{-1.1267}$	$^\dagger\mathbf{-1.1304}$

GENERALISATION $-$ $H_0 : \mu_{CRIT}(\tau_g) = \mu_a(\tau_g), a \in \{L_i, R_i, E_1, E_2\}_{i=1}^4$

	L4	L3	L2	L1	R1	R2	R3	R4	E1	E2
ADJUSTED p-VALUE	$^*\mathbf{0.0409}$	$^*\mathbf{0.0018}$	$^*\mathbf{0.0075}$	$^*\mathbf{0.0409}$	0.3975	$^*\mathbf{0.0296}$	$^*\mathbf{1.03 \times 10^{-5}}$	$^*\mathbf{6.88 \times 10^{-14}}$	$^*\mathbf{0}$	$^*\mathbf{0}$
EFFECT SIZE	-0.1144	-0.0416	-0.0074	0.0039	0.0035	0.0093	0.0128	-0.0113	$^\dagger\mathbf{-1.2549}$	$^\dagger\mathbf{-1.2575}$

Table 6: **Dropout** $-\ \theta = 0.5$: Post-hoc tests for training speed and generalisation for dropout with rate $\theta = 0$. The symbols * indicate a significant p-value and † a large effect size.

TRAINING SPEED $-$ $H_0 : \mu_{CRIT}(\tau_s) = \mu_a(\tau_s), a \in \{L_i, R_i, E_1, E_2\}_{i=1}^4$

	L4	L3	L2	L1	R1	R2	R3	R4	E1	E2
ADJUSTED p-VALUE	0.1170	$^*\mathbf{7.46 \times 10^{-5}}$	$^*\mathbf{0.0075}$	0.0843	0.1188	$^*\mathbf{0.0002}$	$^*\mathbf{1.63 \times 10^{-8}}$	$^*\mathbf{2.66 \times 10^{-14}}$	$^*\mathbf{0}$	$^*\mathbf{0}$
EFFECT SIZE	-0.0491	0.01421	0.0105	0.0119	-0.0227	-0.0073	-0.0210	-0.0404	$^\dagger\mathbf{-1.1158}$	$^\dagger\mathbf{-1.1171}$

GENERALISATION $-$ $H_0 : \mu_{CRIT}(\tau_g) = \mu_a(\tau_g), a \in \{L_i, R_i, E_1, E_2\}_{i=1}^4$

	L4	L3	L2	L1	R1	R2	R3	R4	E1	E2
ADJUSTED p-VALUE	$^*\mathbf{0.0009}$	$^*\mathbf{6.18 \times 10^{-6}}$	$^*\mathbf{0.0016}$	0.0591	0.2637	$^*\mathbf{0.0010}$	$^*\mathbf{1.03 \times 10^{-5}}$	$^*\mathbf{5.08 \times 10^{-12}}$	$^*\mathbf{0}$	$^*\mathbf{0}$
EFFECT SIZE	-3.37×10^{-2}	4.44×10^{-3}	1.01×10^{-5}	8.92×10^{-3}	-2.04×10^{-2}	4.63×10^{-4}	-1.18×10^{-2}	$-2.64e-02$	$^\dagger\mathbf{-1.2796}$	$^\dagger\mathbf{-1.2843}$

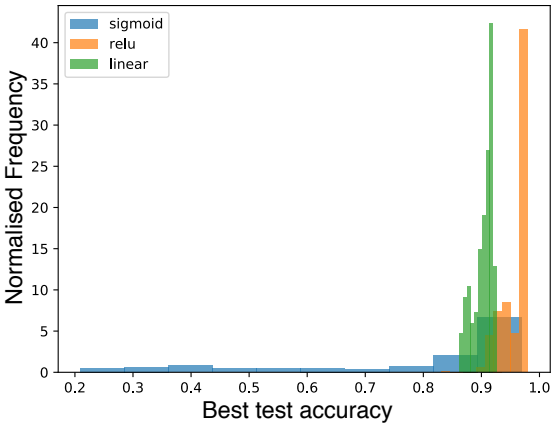


Figure 5: *Best test accuracy distribution per activation function.* The test RCT confirms the validity of this setup as it clearly confirms well known results: (1) ReLU networks typically outperform linear and sigmoidal networks, (2) the best performing sigmoidal networks outperform linear networks and perform comparatively to ReLU networks, but are typically more difficult to train due to vanishing gradients.

to the above. All networks make use of the ReLU activation function and the interventions in this case are initialisation schemes: He (He et al., 2015), Xavier (Glorot and Bengio, 2010) and Orthogonal (Saxe et al., 2014). One of these RCTs makes use of learning rate decay and the other does not. In this way, we can compare the statistical findings of each and confirm whether they agree. If the test results do agree, it means that the RCT has successfully controlled for the confounding effects of learning rate in both cases, i.e. with and without decay. Figure 6 shows best test accuracy distributions without and with learning rate decay. It is clear that although learning rate decay may improve the overall performance of all groups, the relative performance differences between groups remain roughly the same. Table 7 gives the test results for each RCT. The conclusions closely match between the two trials. Therefore, we conclude that the RCT as described and performed in the main text provides a very general approach to isolating the effects of a particular intervention.

Stellenbosch University https://scholar.sun.ac.za

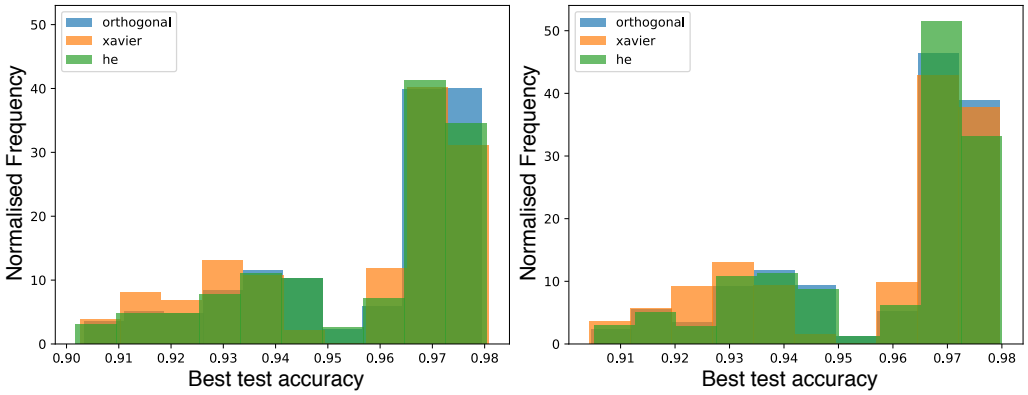


Figure 6: *Comparison of RCTs with and without learning rate decay.* **Left**: Without learning rate decay. **Right**: With learning rate decay. Best test accuracy distributions per initialisation are shown. It is clear that although learning rate decay improves the overall performance of all groups, the relative performance between groups remains the same.

Table 7: **Statistical tests using learning rates with and without decay**: Post-hoc tests for generalisation using and not using learning rate decay. Comparisons are between He and Xavier and He and orthogonal initialisation with the null hypothesis H_0 of no difference. The relative differences as detected by the tests remain the same between the two approaches, thus the RCT has successfully isolated only the effects of the initialisation.

Generalisation		
With decay	Orthogonal	Xavier
Adjusted p-value	0.1972	***0.0904**
Effect size	0.0124	−0.1176
Without decay	Orthogonal	Xavier
Adjusted p-value	0.9183	***0.0067**
Effect size	0.0156	−0.1130

Stellenbosch University https://scholar.sun.ac.za

4.3 Discussion

In **this chapter**, we investigated the effect of off-critical initialisation on training speed and generalisation performance for noise regularised ReLU neural networks of limited depth. A statistical analysis of a large body of empirical data revealed that these networks are actually fairly insensitive to initialisation inside a wide neighbourhood around the critical initialisation (derived in Chapter 3). In other words, there exists a large number of alternative initialisations that perform just as well as the critical initialisation. Given the scale of our study, we find this observation of low sensitivity somewhat surprising: even networks of only moderate depth still pose complicated and non-trivial optimisation problems. However, the findings do seem to align with recent results arguing that the loss landscape of a neural network contains many minima of a similar quality (e.g. due to weight space symmetry) (Brea *et al.*, 2019), and that these minima are reachable from a wide range of starting locations (Draxler *et al.*, 2018). This is in contrast to deep linear neural networks, which only possess a single global minimum (Laurent and von Brecht, 2017) (with all other fixed points corresponding to saddle points) which seem to make learning more dependent on initialisation.

An issue of contention in this work is that of an interaction between learning rate and initialisation. For example, a possible alternative outcome could have been that initialisations do differ, as long as for each initialisation the maximum stable learning rate was used. In the appendix of the paper, we discuss how the RCT still controls for dynamic learning rate schedulers. Furthermore, the grid over which learning rates were sampled in the original study was fairly coarse, but reasonable and showed no causal dependence. Therefore, even though we cannot guarantee that tuned learning rates would change our findings (which could be an interesting follow up study), we find it somewhat unlikely.

Finally, we note that although the analysis in this chapter points towards the conclusion that optimising over the scale of the sampling distribution for i.i.d. sampled weights is perhaps unnecessary, it does not exclude the potential usefulness of other classes of initialisation strategies such as orthogonal, sparse or correlated weight initialisation.

So far, we have explored learning and initialisation for noise regularised neural networks in the context of maximum likelihood inference. Maximum likelihood is an effective approach for building predictive models. However, by only providing a single point estimate of the parameters, maximum likelihood is more susceptible to overfitting and has no mechanism for quantifying the uncertainty associated with a prediction. An alternative approach is to not only model the mean of the assumed data distribution, but also higher order moments that characterise the *full* distribution. In this way, predictions can be averaged to help prevent overfitting and estimates of prediction uncertainty can be obtained using the variance of the distribution. This is the approach taken in *Bayesian inference*.

In the **next chapter**, we venture beyond maximum likelihood and investigate the effects of noise regularisation on neural network models that perform exact Bayesian inference. Our investigation begins with a connection that exists between deep neural networks and *Gaussian processes* (Williams and Rasmussen, 2006).

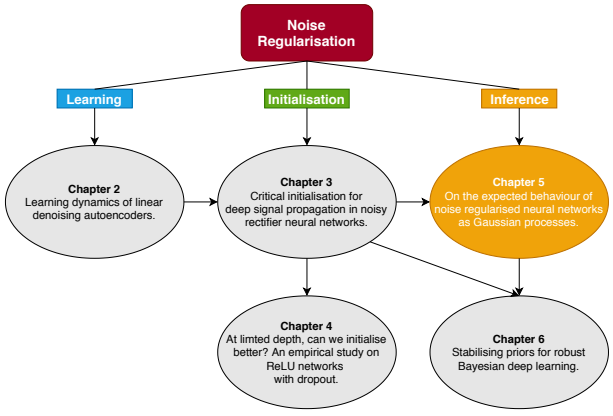
Stellenbosch University https://scholar.sun.ac.za

Chapter 5

Noise regularised neural networks as Gaussian processes

Paper
Pretorius, A., Kamper, H., & Kroon, S. On the expected behaviour of noise regularised neural networks as Gaussian processes. *Preprint (arXiv, 2019).*
Available at: https://arxiv.org/pdf/1910.05563.pdf

Code
Pretorius, A. (2019). Code for: On the expected behaviour of noise regularised neural networks as Gaussian processes.
Available at: https://github.com/arnupretorius/noisyNNGPs_2019



5.1 Introduction

A seminal result in machine learning is that a shallow infinitely wide neural network is equivalent to a *Gaussian process* (GP), where the GP kernel parameters depend on the initialisation of the network (Neal, 1994). Recently, this result has been extended to deep neural networks (Lee *et al.*, 2018; Matthews *et al.*, 2018). Furthermore, Lee *et al.* (2018) showed that these neural network equivalent GPs perform best when their kernel parameters are set equal to their corresponding critical neural network initialisations.

In Chapter 3, we derived critical initialisations for noise regularised neural networks. Therefore, we naturally ask the following question: if we can show an equivalence between noise regularised deep neural networks and GPs, would these initialisation strategies also

Stellenbosch University https://scholar.sun.ac.za

correspond to the optimal settings for a noisy GP kernel. In addition, we are interested in understanding how the kernel, and by extension the covariance of the GP, will behave under noise injection.

In this chapter, we establish an equivalence between deep noise regularised neural networks and GPs. We focus on fully-connected feedforward neural networks with ReLU activations and analyse the behaviour of these models in expectation over the injected noise. We find that the best performing GPs are those that have their kernel parameters set to the values specified by the initialisation derived in Chapter 3. Furthermore, we show how noise regularisation in a neural network affects the covariance of the corresponding noisy GP. As more noise is injected into the network the covariance becomes closer to diagonal, corresponding to complete independence between outputs for training points. This leads to a stronger prior for simple functions and larger uncertainty in the posterior predictive distribution when performing exact Bayesian inference.

5.2 Background

Bayesian inference is a different modeling paradigm to maximum likelihood estimation (outlined in Chapter 1). In Bayesian inference, we model the entire conditional distribution over the outputs given the inputs by modeling the parameters θ in the function $f(\mathbf{x}, \theta)$ with a distribution. In this way, the Bayesian approach encodes uncertainty into the modeling process by not relying on a single point estimate θ^* for the optimal parameters, as is done in maximum likelihood.

In more detail, suppose we are given a set of observations $\mathcal{D} = \{(\mathbf{x}_i, y_i), i = 1, ..., N\}$. The Bayesian approach is to incorporate our prior belief concerning the model parameters by specifying a *prior distribution* $p(\theta)$ over the parameters. This initial belief is updated based on observed samples using *Bayes' rule* to obtain a *posterior distribution* over the parameters of the form

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{\int p(\mathcal{D}|\theta)p(\theta)d\theta}. \tag{5.2.1}$$

We can use this posterior over the weights to model the conditional distribution for a new test point $\mathbf{x}^*$ as follows

$$p(y^*|\mathbf{x}^*, \mathcal{D}) = \int p(y^*|\mathbf{x}^*, \theta)p(\theta|\mathcal{D})d\theta. \tag{5.2.2}$$

Gaussian processes (GPs) are powerful models that perform exact Bayesian inference. GPs model the output by specifying a joint Gaussian distribution over functions evaluated on the training points and condition on this Gaussian to perform inference on new given test points. GPs are non-parametric in the sense that they do not explicitly rely on model parameters θ, but rather depend on a *kernel*. The kernel measures the degree of similarity between function values at points and can be parameterised in various ways. The covariance of the GP is defined by the kernel evaluated at all pairs of points. Inference in GPs is the function space equivalent to exact Bayesian inference in parameter space: the process of conditioning on the training points is implicitly integrating over the posterior of the parameters associated with some parameterised model (Bishop, 2006).

Stellenbosch University https://scholar.sun.ac.za

5.3 Contribution statement

For this paper, I derived the theoretical results, performed the experiments and wrote the paper. Dr Steve Kroon provided technical feedback. Dr Herman Kamper provided general feedback and useful editorial suggestions.

Stellenbosch University https://scholar.sun.ac.za

On the expected behaviour of noise regularised deep neural networks as Gaussian processes

Arnu Pretorius
Computer Science Division
Stellenbosch University

Herman Kamper
Department of E&E Engineering
Stellenbosch University

Steve Kroon
Computer Science Division
Stellenbosch University

Abstract

Recent work has established the equivalence between deep neural networks and Gaussian processes (GPs), resulting in so-called *neural network Gaussian processes* (NNGPs). The behaviour of these models depends on the initialisation of the corresponding network. In this work, we consider the impact of noise regularisation (e.g. dropout) on NNGPs, and relate their behaviour to signal propagation theory in noise regularised deep neural networks. For ReLU activations, we find that the best performing NNGPs have kernel parameters that correspond to a recently proposed initialisation scheme for noise regularised ReLU networks. In addition, we show how the noise influences the covariance matrix of the NNGP, producing a stronger prior towards simple functions away from the training points. We verify our theoretical findings with experiments on MNIST and CIFAR-10 as well as on synthetic data.

1 Introduction

Modern deep neural networks (NNs) are powerful tools for modeling highly complex functions. However, deep NNs lack natural ways of incorporating uncertainty estimation, and (approximate) Bayesian inference for NNs remains a challenge. In contrast, non-parameteric approaches such as Gaussian Processes (GPs) provide exact Bayesian inference and well-calibrated uncertainty estimates, but typically consider substantially simpler models than deep NNs. Therefore, a large body of work has recently emerged attempting to combine parametric deep learning models and GPs so as to derive benefits from both. These approaches include deep GPs (Damianou and Lawrence, 2013; Duvenaud et al., 2014; Hensman and Lawrence, 2014; Dai et al., 2015; Bui et al., 2016; Salimbeni and Deisenroth, 2017), deep kernel learning (Wilson et al., 2016a,b; Al-Shedivat et al., 2016) and viewing deep learning with dropout as an approximate deep GP (Gal and Ghahramani, 2016).

For shallow infinite width neural networks, an exact equivalence to GPs has been known for some time (Neal, 1994; Williams, 1997; Le Roux and Bengio, 2007). However, this equivalence has only recently been extended to deeper architectures (Hazan and Jaakkola, 2015; Lee et al., 2018; Matthews et al., 2018; Novak et al., 2019). Referred to as *neural network Gaussian processes* (NNGPs) in Lee et al. (2018), the resulting models are GPs with an exact correspondence to infinitely wide deep neural networks. Importantly, the NNGP depends on the hyperparameters of the network and its initialisation, which determines the network's signal propagations dynamics.

In deep neural networks, signal propagation has been shown to exhibit distinct phases depending on the initialisation of the network (Poole et al., 2016). These phases include ordered and chaotic regimes associated with vanishing and exploding gradients respectively, which can result in poor network performance (Schoenholz et al., 2017). By initialising at the critical boundary between these two regimes, known as the "edge of chaos", the flow of information through the network improves, often resulting in faster and deeper training for a variety of architectures (Pennington et al., 2017; Yang and Schoenholz, 2017; Chen et al., 2018; Xiao et al., 2018).

Lee et al. (2018) showed that NNGPs tend to inherit the above behaviour from their corresponding randomly initialised networks. In particular, there exists an interaction between poor signal propagation and a poorly constructed kernel. As a result, the performance of NNGPs tend to suffer if their kernels are constructed using kernel parameters that correspond to network

Stellenbosch University https://scholar.sun.ac.za

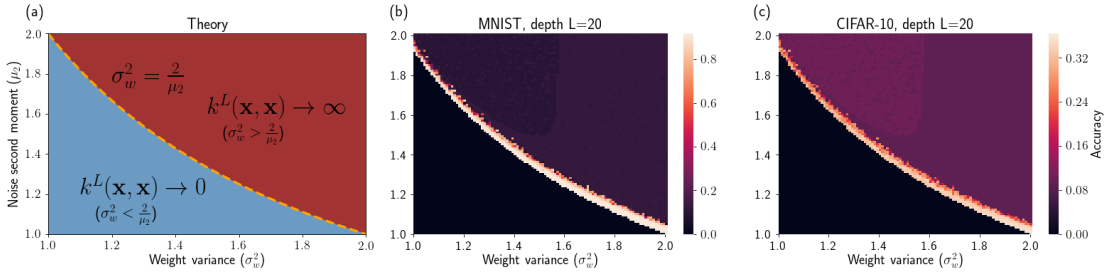


Figure 1: *Dependence of noisy NNGPs on critical parameters for performance.* **(a)** Critical boundary for kernel parameters $\{\sigma_w^2, \sigma_b^2\} = \{2/\mu_2, 0\}$ as a function of noise. **(b)** MNIST test accuracy for a 20-layer noisy NNGP, for kernel parameters $\sigma_b^2 = 0$, $\sigma_w^2, \mu_2 \in [1.0, 2.0]$ (both sampled in interval sizes of 0.01). Training and test set sizes are $N = 1000$. **(c)** CIFAR-10 test accuracy, details same as (b).

initialisations far from the critical boundary. Furthermore, even at the critical boundary, inputs to a neural network can still become asymptotically correlated at large depth (Schoenholz et al., 2017). The rate of convergence in this correlation limits the depth to which networks can be trained, because after this convergence the network is unable to distinguish between different training observations at the output layer. This dependence on depth (in the constructed kernel) for performance, also manifests in NNGPs (Lee et al., 2018).

Various design decisions are required to instantiate a modern NN. Important decisions for trainability and test performance often include both initialisation *and* regularisation. If initialised poorly, a network might become untrainable due to poor signal propagation, whereas a lack of regularisation could hurt the test performance of the network if it starts to overfit. Commonly used approaches to alleviating these issues include principled initialisation schemes (Glorot and Bengio, 2010; He et al., 2015) and improved regularisation strategies. Among the most successful regularisation strategies is dropout (Srivastava et al., 2014), a form of noise regularisation where scaled Bernoulli noise is applied multiplicatively to the units of a network to prevent co-adaptation. However, as shown by Pretorius et al. (2018), these components do not act in isolation and therefore the initialisation of the network should depend on the amount of noise regularisation being applied.

In this paper, we investigate the following research question: do the signal propagation dynamics that influence noise regularised NNs also govern the behaviour of corresponding "noisy NNGPs"? In the presence of multiplicative noise regularisation, Pretorius et al. (2018) derived the critical initialisation for stable signal propagation in feedforward ReLU networks: More specifically, the authors showed that stable propagation is achieved by setting all unit biases to zero and sampling the weights from zero mean Gaussians with variance σ_w^2/D_{in} set equal to $\sigma_w^2 = 2/\mu_2$. Here, D_{in} is the number of incoming units to the layer and μ_2 is the second moment of the noise. For example, when using dropout, $\mu_2 = 1/p$ (where p is the probability of keeping the unit active) and therefore the initial weights are sampled from $\mathcal{N}(0, 2p/D_{\text{in}})$. Furthermore, it was shown that the rate of convergence to a fixed point correlation between inputs increases as a function of the amount of regularisation being applied. Consequently, increased noise further limits the depth of trainability in neural networks. In this paper, we investigate whether these findings for noise regularised networks carry across to their noisy NNGP counterparts.

We consider noise regularised fully-connected feedforward NNs and study the behaviour of noisy NNGPs. Our analysis is done in expectation over the noise, under a general noise model (of which dropout is a special case). We give the kernel corresponding to noisy ReLU NNGPs and highlight the different noise-induced degeneracies in the kernel as the depth becomes large. Specifically, we show that the above noise dependent initialisations promoting stable signal propagation in noisy ReLU NNs correspond exactly to the kernel parameters exhibiting good performance in noisy NNGPs, as shown in Figure 1. However, even at criticality, we show that as the noise tends to infinity the covariance of the NNGP becomes diagonal. As a result, noise regularisation translates into a stronger prior for simple functions away from the training points. Finally, we verify our findings with experiments on real-world and synthetic datasets.

2 Noise regularised deep neural networks as Gaussian processes

We consider a noise regularised fully-connected deep feedforward neural network. Given an input $\mathbf{x}^0 \in \mathbb{R}^{D_0}$, we inject noise into the model

$$\tilde{\mathbf{h}}^l = W^l(\mathbf{x}^{l-1} \odot \epsilon^{l-1}) + \mathbf{b}^l, \qquad \text{for } l = 1, ..., L \quad (1)$$

Stellenbosch University https://scholar.sun.ac.za

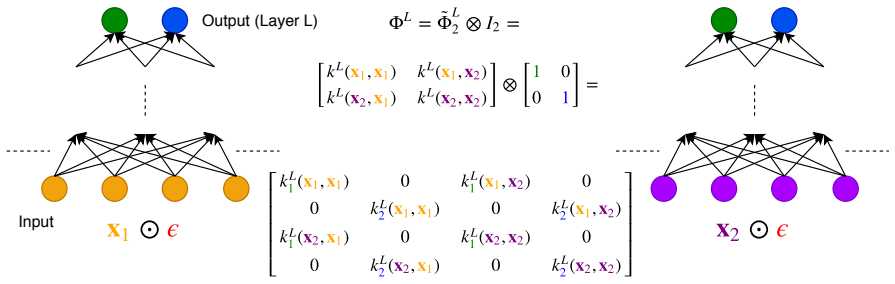


Figure 2: *Noisy NNGP covariance*: Example of the covariance for a noisy NNGP with only two inputs $\mathbf{x}_1$ (orange), $\mathbf{x}_2$ (purple) and two output units (green and blue) at layer L and sampled noise ϵ.

using the operator $\odot$ to denote either addition or multiplication, where ϵ^l is an input noise vector, sampled from a pre-specified noise distribution. The noise is assumed to have $\mathbb{E}[\epsilon^l] = 0$ in the additive case, and $\mathbb{E}[\epsilon^l] = 1$ for multiplicative noise distributions such that in both cases, $\mathbb{E}_\epsilon[\tilde{\mathbf{h}}^l] = W^l \mathbf{x}^{l-1} + \mathbf{b}^l$. The weights $W^l \in \mathbb{R}^{D_l \times D_{l-1}}$ and biases $\mathbf{b}^l \in \mathbb{R}^{D_l}$ are sampled i.i.d. from zero mean Gaussian distributions with variances σ_w^2/D_{l-1} and σ_b^2, respectively, where D_l denotes the dimensionality of the l^{th} hidden layer in the network. Each hidden layer's activations $\mathbf{x}^l = \phi(\tilde{\mathbf{h}}^l)$ are computed element-wise using an activation function $\phi(\cdot)$. Lastly, we denote the second moment of the noise as $\mu_2 = \mathbb{E}[\epsilon^2]$ and define $\tilde{\mathbf{h}}^L = f(\mathbf{x})$ as the entire function mapping from input to output, with $\mathbf{x} = \mathbf{x}^0$.

By choosing parameter sampling distributions at initialisation we are implicitly specifying a prior over networks in parameter space. We now transition from parameter space to function space by instead specifying a prior directly over function values. Assume a training set of input-output pairs $\mathcal{D} = \{(\mathbf{x}_i, y_i), i = 1, ..., N\}$. If we can show that $\mathbf{f} = [f(\mathbf{x}_1), ..., f(\mathbf{x}_N)]^T$ is Gaussian distributed at initialisation, then the distribution over the output of the network at these points is completely determined by the second-order statistics $\mathbb{E}[\mathbf{f}] = \mathbf{m}$ and $\text{Cov}[\mathbf{f}] = K$, defining the following GP

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{m}, K) \equiv \mathcal{GP}(\mathbf{m}, K). \tag{2}$$

We begin by assuming the following additive error model with regression outcomes $y_i = f(\mathbf{x}_i) + \varepsilon_i$, where $\varepsilon_i \sim \mathcal{N}(0, \sigma_\varepsilon^2)$.[1] Since $y_i|\mathbf{x}_i \sim \mathcal{N}(f(\mathbf{x}_i), \sigma_\varepsilon^2)$, the joint distribution over all outcomes is

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{f}, \sigma_\varepsilon^2 \mathbf{I}_N), \tag{3}$$

where $\mathbf{y} = [y_1, ..., y_N]^T$. In GP regression we are interested in finding the marginal distribution

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}) d\mathbf{f}. \tag{4}$$

We proceed as in (Lee et al., 2018) to argue that $\mathbf{f}$ is in fact a zero mean Gaussian (we refer the reader to Matthews et al. (2018) for a more formal approach) and derive the elements of the covariance matrix K in (2) for noise regularised deep neural networks. Subsequently, we obtain an expression for (4) by combining (2) and (3) and using standard results for the marginal of a Gaussian.

To show that the expected distribution of $\mathbf{f}$ over the injected noise is Gaussian, we first note that conditioned on the inputs, the "output" units at layer l, stemming from the post-activations $\mathbf{x}^{l-1}$ in the previous layer are given by $\tilde{h}_d^l = \mathbf{w}_d^l \cdot (\mathbf{x}^{l-1} \odot \epsilon^{l-1}) + b_d^l$, for $d = 1, ..., D_l$. As previously mentioned, we sample the weights and biases i.i.d. from a zero mean Gaussian and define the noise to be i.i.d. such that $\tilde{h}_d^l$ is unbiased in expectation of the injected noise. Therefore, in a wide network, $\tilde{h}_d^l$ is a sum of a large collection of i.i.d. random variables. As $D_{l-1} \to \infty$, the central limit theorem ensures that the distribution of $\tilde{h}_d^l$ will tend to a Gaussian with mean $\mathbb{E}[\tilde{h}_d^l(\mathbf{x}_i)] = 0$ and covariance $\mathbb{E}\left[\tilde{h}_d^l(\mathbf{x}_i)\tilde{h}_d^l(\mathbf{x}_j)\right]$. As a result, the function values $h_d^l(\mathbf{x}_1), ..., h_d^l(\mathbf{x}_N), \forall d$ can be treated as samples from a GP given by $\tilde{\mathbf{h}}^l \sim \mathcal{GP}(\mathbf{0}, \Phi^l)$. Here, Φ^l is an $ND_l \times ND_l$ covariance matrix given by

$$\begin{aligned}\Phi^l &= \begin{bmatrix} k^l(\mathbf{x}_1, \mathbf{x}_1) & k^l(\mathbf{x}_1, \mathbf{x}_2) & \dots & k^l(\mathbf{x}_1, \mathbf{x}_N) \\ k^l(\mathbf{x}_2, \mathbf{x}_1) & k^l(\mathbf{x}_2, \mathbf{x}_2) & \dots & k^l(\mathbf{x}_2, \mathbf{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ k^l(\mathbf{x}_N, \mathbf{x}_1) & k^l(\mathbf{x}_N, \mathbf{x}_2) & \dots & k^l(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix} \otimes I_{D_l} \\ &= \tilde{\Phi}_N^l \otimes I_{D_l},\end{aligned}$$

where $\otimes$ is the Kronecker product. The kernel function $k^l(\mathbf{x}_i, \mathbf{x}_j) \equiv \mathbb{E}\left[\tilde{h}_\cdot^l(\mathbf{x}_i)\tilde{h}_\cdot^l(\mathbf{x}_j)\right]$ depends on the layer depth, the scale of the weights and biases and the amount of noise regularisation being applied. A schematic illustration of the covariance matrix is given

[1] Note that here we consider scalar outputs, i.e. $f : \mathbb{R}^{D_0} \to \mathbb{R}$, hence $\tilde{h}^L = f(\mathbf{x}) \in \mathbb{R}$. Also, the additive error noise ε should not be confused with the injected noise ϵ^l in (1).

Stellenbosch University https://scholar.sun.ac.za

in Figure 2 for the simple case of only two inputs and two output units. To derive the elements of the covariance Φ^l, consider the units $\tilde{h}^l_d, \tilde{h}^l_s$, $d, s \in \{1, ..., D_l\}$ and inputs $\mathbf{x}_i, \mathbf{x}_j, i, j \in \{1, ..., N\}$ which give

$$k^l_{ds}(\mathbf{x}_i, \mathbf{x}_j) = \tilde{\Phi}^l_{ij} \otimes I_{ds} = \begin{cases} \mathbb{E}\left[\tilde{h}^l_d(\mathbf{x}_i)\tilde{h}^l_s(\mathbf{x}_j)\right] & , \text{ if } d = s \\ 0 & , \text{otherwise.} \end{cases}$$

Note that $k^l_{ds}(\mathbf{x}_i, \mathbf{x}_j) = 0, \forall i, j$ and $d \neq s$ due to the independence between the incoming connections (weights) associated with each output unit. Therefore, we only consider the case where $d = s$, which for $i \neq j$ gives

$$\begin{aligned} k^l_d(\mathbf{x}_i, \mathbf{x}_j) &= \mathbb{E}_{\mathbf{w},b,\epsilon}\left[\tilde{h}^l_d(\mathbf{x}_i)\tilde{h}^l_d(\mathbf{x}_j)\right] \\ &= \frac{\sigma^2_w}{D_{l-1}} \sum_{d'=1}^{D_{l-1}} \left[\phi\left(\tilde{h}^{l-1}_{d'}(\mathbf{x}_i)\right)\phi\left(\tilde{h}^{l-1}_{d'}(\mathbf{x}_j)\right)\right] + \sigma^2_b \\ &= \sigma^2_w \mathbb{E}\left[\phi\left(\tilde{h}^{l-1}_{d'}(\mathbf{x}_i)\right)\phi\left(\tilde{h}^{l-1}_{d'}(\mathbf{x}_j)\right)\right] + \sigma^2_b, \end{aligned}$$

where the expectation is taking with respect to $\tilde{\mathbf{h}}^{l-1} \sim \mathcal{GP}(\mathbf{0}, \Phi^{l-1})$. The final equality follows from applying the above argument recursively for the previous layer $l-1$. For the case of $i = j$ (and $d = s$), we have that the diagonal components of the covariance matrix are given by

$$k^l_d(\mathbf{x}_i, \mathbf{x}_i) = \sigma^2_w \left\{\mathbb{E}\left[\phi\left(\tilde{h}^{l-1}_{d'}(\mathbf{x}_i)\right)^2\right] \odot \mu_2\right\} + \sigma^2_b,$$

where the influence of the noise ϵ is explicitly expressed through its second moment μ_2. Using the initial condition

$$k^0(\mathbf{x}_i, \mathbf{x}_j) = \mathbb{E}_\epsilon[(\mathbf{x}_i \odot \epsilon) \cdot (\mathbf{x}_j \odot \epsilon)]$$

and letting each layer width in the network $D_1, ..., D_L$ tend to infinity in succession, this recursive construction gives $\mathbf{f}$ as Gaussian distributed with mean and covariance

$$\mathbf{m} = \mathbf{0}, \qquad K = \Phi^L. \tag{5}$$

Finally, combining (2), (3) and (5) and using standard results for the marginal of a Gaussian distribution, the marginal in (4) can be shown to be

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{0}, \Psi),$$

where $\Psi(\mathbf{x}_i, \mathbf{x}_j) = K_{ij} + \sigma^2_\varepsilon \delta_{ij}$ with δ_{ij} the Kronecker delta (Williams and Rasmussen, 2006). Therefore, together with the additive noise level σ^2_ε, our model for the joint distribution over training outcomes is fully determined by the equivalent kernel corresponding to layer-wise recursion of an infinite basis function expansion. This kernel, in turn, depends on the parameterisation of the network and the amount of *injected* noise.

Having developed our noisy NNGP model, we next discuss predicting outcomes for unseen test data points. To make a prediction, we evaluate the predictive distribution $p(y^*|\mathbf{x}^*, \mathcal{D})$ at a new test point $\mathbf{x}^*$. Consider the joint

$$p(\mathbf{y}, y^*|\mathbf{f}, \mathbf{x}^*) = \mathcal{N}(\mathbf{0}, \Psi^*)$$

where we can partition the covariance Ψ^* as follows

$$\Psi^* = \begin{bmatrix} \Psi & \mathbf{k} \\ \mathbf{k}^T & \psi \end{bmatrix}$$

with $\mathbf{k} = [k^L_1(\mathbf{x}_1, \mathbf{x}^*), ..., k^L_1(\mathbf{x}_N, \mathbf{x}^*), ..., k^L_{D_L}(\mathbf{x}_N, \mathbf{x}^*)]^T$, an ND_L dimensional vector and $\psi = k^L(\mathbf{x}^*, \mathbf{x}^*) + \sigma^2_\varepsilon$. Using standard results for the conditional distribution of a partitioned Gaussian vector, we find

$$p(y^*|\mathbf{f}, \mathbf{x}^*, \mathbf{y}) = \mathcal{N}(\mu(\mathbf{x}^*), \sigma^2(\mathbf{x}^*)) \tag{6}$$

where $\mu(\mathbf{x}^*) = \mathbf{k}^T\Psi^{-1}\mathbf{y}$ and $\sigma^2(\mathbf{x}^*) = \psi - \mathbf{k}^T\Psi^{-1}\mathbf{k}$. This result is the function space equivalent to exact Bayesian inference in parameter space: by computing the conditional in (6), we are implicitly performing an integration over the posterior of the parameters associated with an infinitely wide noise regularised neural network (Williams, 1997).

In the next section, we study how the properties of the kernel derived in this section depend the parameters of the network when using ReLU activations. Furthermore, for the remainder of the paper we drop the dependence on the hidden units and training set indices for ease of notation and simply refer to $k^l_d(\mathbf{x}_i, \mathbf{x}_j)$ as $k^l(\mathbf{x}, \mathbf{x}')$.

3 Kernel parameters and critical neural network initialisation

We begin by examining the interaction between the parameters of the noisy NNGP kernel and its corresponding network initialisation. Specifically, we focus on ReLU activations and show that the kernel parameter values that lead to non-degenerate kernels for deep noisy NNGPs are exactly those that promote stable signal propagation in noise regularised ReLU networks.

Let $\phi(a) = \text{ReLU}(a) = \max(0, a)$ and define

$$\rho^l_{\mathbf{x},\mathbf{x}'} = \frac{k^l(\mathbf{x}, \mathbf{x}')}{\sqrt{k^l(\mathbf{x}, \mathbf{x})k^l(\mathbf{x}', \mathbf{x}')}},$$

then the elements of the covariance Φ^l at a hidden unit are

$$k^l(\mathbf{x}, \mathbf{x}') = \frac{\sigma^2_w}{2} k^{l-1}(\mathbf{x}, \mathbf{x}')\left\{g(\rho^{l-1}_{\mathbf{x},\mathbf{x}'}) + \frac{1}{2}\right\} + \sigma^2_b, \tag{7}$$

Stellenbosch University https://scholar.sun.ac.za

Table 1: Limiting behaviour for degenerate and critical noisy ReLU kernels.

Weight variance	Bias variance	Limiting value as $L \to \infty$
- Additive Noise ($\mu_2 > 0$) -		
A.1) $0 \leq \sigma_w^2 < 2$	$\sigma_b^2 \geq 0$	$k^L(\mathbf{x}, \mathbf{x}) \to a$ (const. independent of $\mathbf{x}$)
A.2) $\sigma_w^2 \geq 2$	$\sigma_b^2 \geq 0$	$k^L(\mathbf{x}, \mathbf{x}) \to \infty$
- Mult. Noise ($\mu_2 > 1$) -		
M.1) $0 \leq \sigma_w^2 < 2/\mu_2$	$\sigma_b^2 = 0$	$k^L(\mathbf{x}, \mathbf{x}) \to 0$
M.2) $0 \leq \sigma_w^2 < 2/\mu_2$	$\sigma_b^2 > 0$	$k^L(\mathbf{x}, \mathbf{x}) \to a$ (const. independent of $\mathbf{x}$)
M.3) $\sigma_w^2 > 2/\mu_2$	$\sigma_b^2 \geq 0$	$k^L(\mathbf{x}, \mathbf{x}) \to \infty$
M.4) $\sigma_w^2 = 2/\mu_2$	$\sigma_b^2 > 0$	$k^L(\mathbf{x}, \mathbf{x}) \to \infty$
M.5) $\sigma_w^2 = 2/\mu_2$	$\sigma_b^2 = 0$	$k^L(\mathbf{x}, \mathbf{x}) \to k^{L-1}(\mathbf{x}, \mathbf{x}) = ... = k^0(\mathbf{x}, \mathbf{x})$

for $\mathbf{x} \neq \mathbf{x}'$, where

$$g(\rho^l_{\mathbf{x},\mathbf{x}'}) = \frac{\rho^l_{\mathbf{x},\mathbf{x}'} \sin^{-1}\left(\rho^l_{\mathbf{x},\mathbf{x}'}\right) + \sqrt{1 - (\rho^l_{\mathbf{x},\mathbf{x}'})^2}}{\pi \rho^l_{\mathbf{x},\mathbf{x}'}},$$

with diagonal elements given by

$$k^l(\mathbf{x}, \mathbf{x}) = \frac{\sigma_w^2}{2} k^{l-1}(\mathbf{x}, \mathbf{x}) \odot \mu_2 + \sigma_b^2. \tag{8}$$

These formulae are the kernel equivalent to the signal propagation recurrences derived in (Pretorius et al., 2018) for noisy ReLU networks. For no noise and outside the context of GPs, a similar result can be found in (Cho and Saul, 2009). Repeated substitution in (8) shows that

$$\begin{aligned} k^L(\mathbf{x}, \mathbf{x}) &= \frac{\sigma_w^2}{2}\left(\frac{\sigma_w^2}{2} k^{L-2}(\mathbf{x}, \mathbf{x}) \odot \mu_2 + \sigma_b^2\right) \odot \mu_2 + \sigma_b^2 \\ &\vdots \\ &= \begin{cases} \left(\frac{\sigma_w^2}{2}\right)^L k^0(\mathbf{x}, \mathbf{x}) + \sum_{l=0}^{L-1} \left(\frac{\sigma_w^2}{2}\right)^l (\mu_2 + \sigma_b^2), \\ \qquad \text{if } \odot \equiv + \text{ (Additive noise)}, \\ \left(\frac{\sigma_w^2 \mu_2}{2}\right)^L k^0(\mathbf{x}, \mathbf{x}) + \sum_{l=0}^{L-1} \left(\frac{\sigma_w^2 \mu_2}{2}\right)^l \sigma_b^2, \\ \qquad \text{if } \odot \equiv \times \text{ (Multiplicative noise)}. \end{cases} \end{aligned} \tag{9}$$

The limiting properties of the kernel are seen by letting $L \to \infty$ in (9). In this limit, several degenerate kernels arise, analogous to cases of unstable signal propagation dynamics in mean-field theory and other related work (Poole et al., 2016; Daniely et al., 2016; Schoenholz et al., 2017; Pretorius et al., 2018). We provide the different cases in Table 1. For any amount of additive noise, all possible settings (see A.1 and A.2) of the kernel parameters σ_w^2 and σ_b^2 in (9) will result in a degenerate kernel in the limit of infinite depth. The situation is similar for multiplicative noise, except for the case (M.5), where $\{\sigma_w^2, \sigma_b^2\} = \{2/\mu_2, 0\}$. We refer to these parameters in (M.5) as the *critical kernel parameters*. Here, the diagonal elements of the covariance stay fixed at their initial values even at extreme depth. These parameter values are identical to the proposed initialisations for deep noisy ReLU networks derived in (Pretorius et al., 2018).

From (7) we can see that the off-diagonal elements of the covariance matrix are influenced by the noise level at the critical values through the relationship $\sigma_w^2/2 = 1/\mu_2$. Furthermore, we note that $k^l(\mathbf{x}, \mathbf{x}') \to 0$ as $\mu_2 \to \infty$. Therefore, multiplicative noise regularisation has a damping effect on the kernel function evaluated between different inputs, tending towards total dissimilarity and a diagonal covariance. This reduction in the richness of the covariance structure exploitable by the NNGP then enforces a strong prior for simple functions away from the training points. To see this effect, consider the predictive distribution in (6), for a test point $\mathbf{x}^*$. For large amounts of noise, $\mathbf{k} \to \mathbf{0}$ and therefore in the limit, $\mu(\mathbf{x}^*) = 0$ and $\sigma^2(\mathbf{x}^*) = \psi$. Since Ψ is symmetric positive definite by definition and $\mathbf{k}^T \Psi^{-1} \mathbf{k} \geq 0, \forall \mathbf{k}$, the predicted outcome y^* will be a sample from a zero mean Gaussian with maximal uncertainty as measured by the variance ψ, i.e $y^* | \mathbf{x}^* \sim \mathcal{N}(0, \psi)$.

To validate the above claims, the following section provides an empirical investigation. In particular, we test two hypotheses that stem from the above theoretical analysis, using both real-world and synthetic datasets.

4 Experiments

We have shown how the kernel parameters for a noisy NNGP relate to those for its corresponding deep neural network. In doing so, our discussion has led us to the following testable hypotheses:

H1- Noisy NNGPs perform best at their critical parameters: The sensitivity of the kernel parameters should cause noisy NNGPs to perform poorly at settings far away from the critical kernel parameters. Furthermore, the reliance on these

Stellenbosch University https://scholar.sun.ac.za

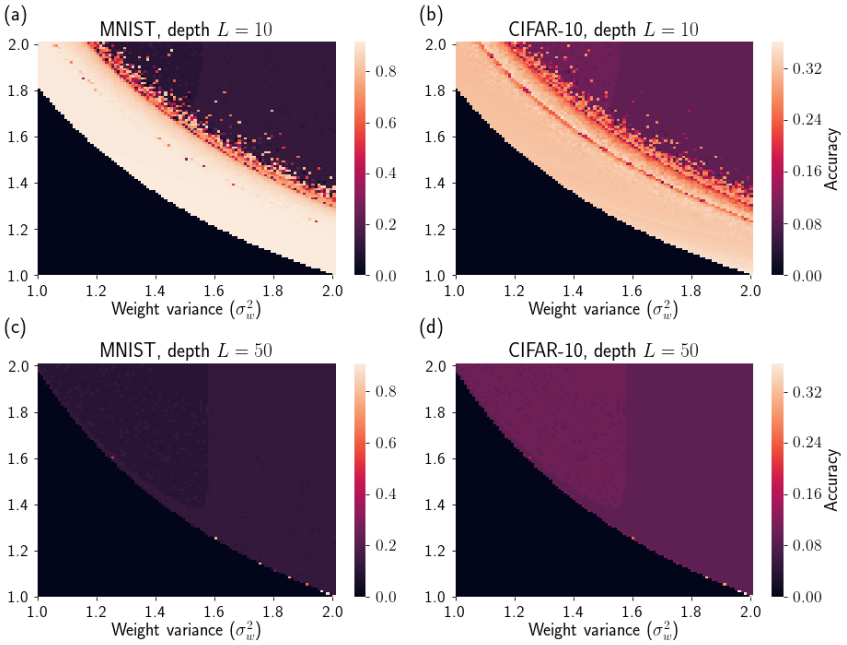


Figure 3: *Sensitivity of NNGP kernel parameters for different depths.* **(a)** Test accuracy for NNGP with depth $L = 10$ on MNIST with training and test set sizes of $N = 1000$ and kernel parameters $\sigma_b^2 = 0$, $\sigma_w^2 = 1.0$ to 2.0 and $\mu_2 = 1.0$ to 2.0, both equally spaced using interval sizes of 0.01. **(b)** NNGP with depth $L = 10$ on CIFAR-10 ($N = 1000$). **(c) - (d)** Same as (a) and (b) but with depth $L = 50$.

critical values for good performance should become more marked at greater depth [Figures 1 and 3].

H2- Noise constrains the covariance and leads to simpler models away from the training points with larger uncertainty: Even at criticality, noise injection applies a shrinkage effect to the kernel function evaluated between different inputs to the noisy NNGP. This should lead to a constrained covariance structure, or in the limit of a large amount of noise, a completely diagonal covariance. The NNGP prior over functions regularised in this way should lead to simpler models away from the training points with increased estimates of uncertainty [Figures 4 and 5].

H1: We begin by investigating the sensitivity of the kernel parameter values. As shown in Figure 1, we test the performance of 20-layer NNGPs on MNIST and CIFAR-10 with kernels constructed for a grid of variance parameters $\sigma_w^2 = 1.0, 1.01, ..., 1.99, 2.0$, for varying values of the noise level parameter $\mu_2 = 1.0, 1.01, ..., 1.99, 2.0$. Our approach to classification in this paper is identical to (Lee et al., 2018), where classification is treated as a regression problem. Specifically, instead of one-hot output vectors, each output vector is recoded as a zero mean regression output with the value 0.9 in the index corresponding to the correct class and -0.1 for all other indices corresponding to the incorrect classes. The predicted class label for a given input is then simply the index corresponding to the maximum value in the output vector as predicted by the NNGP regression model. For all experiments, we set $\sigma_b^2 = 0$. Figures 1(b) and (c) confirm our expectations, showing that the kernel parameters corresponding to NNGPs with good performance closely follow the critical initialisation boundary $\sigma_w^2 = 2/\mu_2$ shown in Figure 1(a). As kernels are constructed further away from criticality, their performances start to deteriorate.

The sensitivity to the kernel parameters becomes more acute at larger depth as shown in Figure 3. Panels (a) and (b) plot the results for a shallower depth of $L = 10$. In this case, a wide band is seen to form around the critical boundary (beige shaded area) with kernel parameters far away from their critical values still able to perform reasonably well. This is no longer the case in Panels (c) and (d), where we tested performances at a greater depth, $L = 50$. At this depth, the NNGP is far more sensitive to the kernel parameters and only a few models with kernel parameters very close to the critical boundary are seen to perform well.

H2: For all the models evaluated in H1, we also study the influence of the noise on the kernel as well as on the resulting NNGP covariance matrix. For each model,

Stellenbosch University https://scholar.sun.ac.za

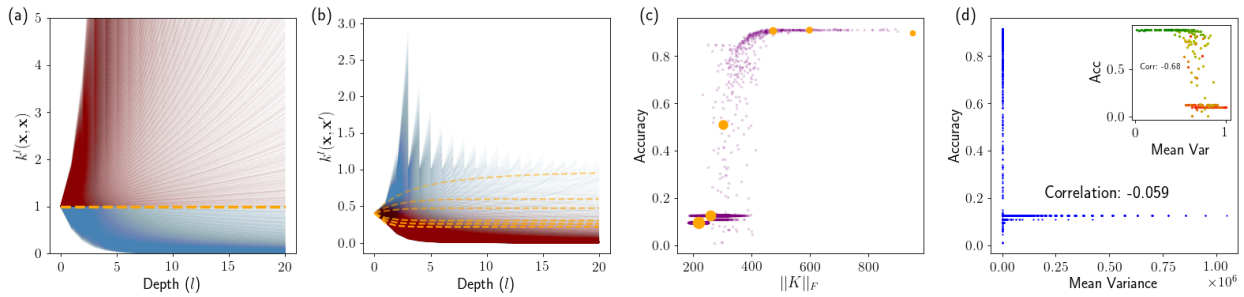


Figure 4: *Effects of noise induced regularisation on the noisy NNGPs in Figure 1 for MNIST.* **(a)** Depth evolution of $k^l(\mathbf{x}, \mathbf{x})$ for the different kernel parameters $\sigma_w^2 = 2/\mu_2$ (dashed orange lines), $\sigma_w^2 > 2/\mu_2$ (solid red lines) and $\sigma_w^2 < 2/\mu_2$ (solid blue lines), with $\sigma_b^2 = 0$ in all experiments. **(b)** Depth evolution of $k^l(\mathbf{x}, \mathbf{x}')$ **(c)** Relationship between accuracy and covariance norm. Orange points correspond to critical kernel parameters with larger sizes indicating more noise. **(d)** Scatter plot of accuracy as a function of mean variance. We measure the quality of uncertainty estimates by computing the correlation of the mean posterior predictive variance with test accuracy. Main plot contains all points, whereas the inset only contains points close to criticality (green to red showing an increase in noise).

we plot in Figure 4(a) and (b) the depth evolution of the kernel, using two inputs from the MNIST dataset. In (a) we track the variance of one of the inputs and in (b) the covariance between the two inputs. The dashed orange lines correspond to the kernel parameters $\sigma_w^2 = 2/\mu_2$, with $\sigma_w^2 > 2/\mu_2$ shown in solid red and $\sigma_w^2 < 2/\mu_2$ shown in solid blue. (Recall that $\sigma_b^2 = 0$ for all experiments). The limiting behaviour described in (M.1), (M.3) and (M.5) in Table 1 is shown in (a), with all kernels tending towards degenerate function mappings, except those evolving under the critical parameters. Furthermore, in (b), we show the damping effect on the kernel at criticality, highlighted by decreasing asymptotes (dashed orange lines around layer 20) as more noise is being applied (darker lines).

The depth dynamics of the kernel also constrains the resulting covariance matrix. To see the effect of this, we use the Frobenius norm of the covariance matrix as a proxy for its richness. Figure 4(c) plots the relationship between the covariance norm and test accuracy for all the experiments in H1. Interestingly, the norm seems to suggest a step function relationship. Moving from right to left in (c), we observe a sudden and dramatic drop in performance beyond a certain amount of regularisation, as measured by a decreasing covariance norm. In other words, there seems to exist some requisite amount of information to be captured by the covariance matrix in order for the NNGP to be able to perform well. This is also the case at criticality: in (c), the orange points correspond to critical kernels with larger points associated with more noise.

The effect of increased noise regularisation on uncertainty estimation is shown in Figure 4(d), where we plot test accuracy as a function of the mean posterior predictive variance. For NNGPs far away from criticality (blue points in main plot), we see little correlation (-0.059) between variance and test accuracy. The inset in (d) shows the same plot but for NNGPs close to their critical parameters. For these models the (negative) correlation is stronger (-0.68), possibly providing more reliable uncertainty estimates. Here, the green points are low noise models and the red points are high noise models. As expected, noise regularisation causes the posterior predictive variance to increase leading to higher uncertainty. We did the same investigations using the CIFAR-10 dataset and obtained similar results (see Appendix A).

Finally, to gain more insight, we consider a simple one-dimensional regression task.[2] The top row in Figure 5 shows samples from a 20-layer NNGP prior for (a) $\mu_2 = 1.0$ (no noise), (b) $\mu_2 = 1.001$ (small noise), (c) $\mu_2 = 2.0$ (large noise) and $\{\sigma_w^2, \sigma_b^2\} = \{2/\mu_2, 0.05\}$. We found a small amount of bias $\sigma_b^2 = 0.05$, improved each fit (see Appendix B for a discussion on non-zero biases). The covariance structure corresponding to each NNGP is shown in (d)-(f), located in the middle row of Figure 5. The bottom row, (g)-(i) shows the fit from the posterior predictive (red line) using four training examples (blue dots) sampled from a simple sinusoidal function (green line) with $\sigma_\varepsilon = 0.1$. Moving across the columns from left to right, we find that the samples from the prior become more erratic as the covariance becomes diagonal, which strongly regularises the regression model at previously unseen test points. Note that the model in (i) still corresponds to exact

[2]The example is taken from Chapter 1 in Bishop (2006).

Stellenbosch University https://scholar.sun.ac.za

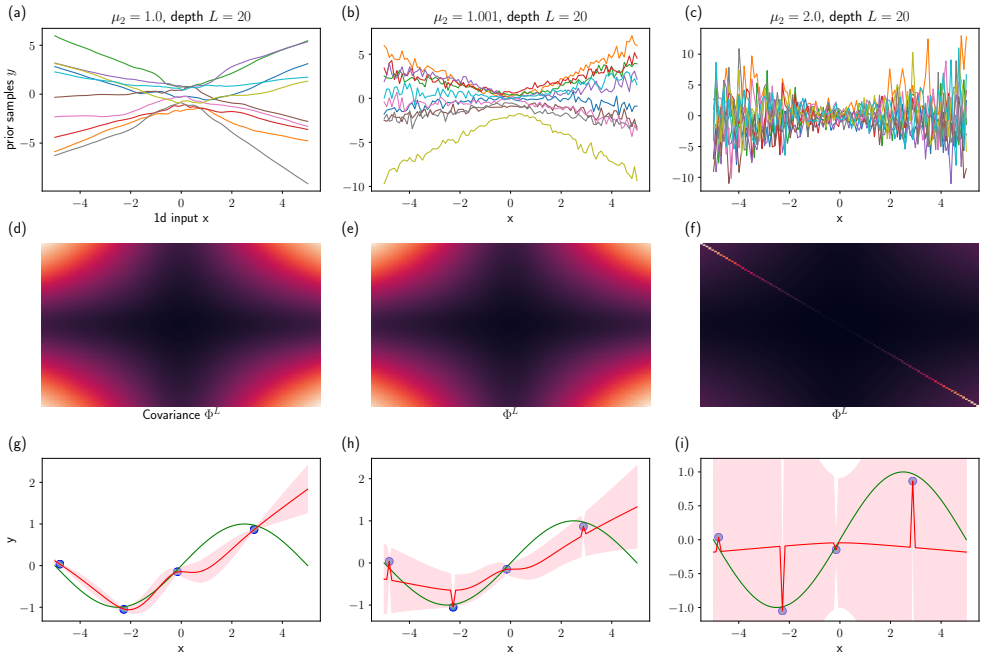


Figure 5: *20-layer noisy NNGPs with 1D input data and* $\{\sigma_w^2, \sigma_b^2\} = \{2/\mu_2, 0.05\}$: Left column $\mu_2 = 1.0$, middle column $\mu_2 = 1.001$ and right column $\mu_2 = 2.0$. **(a) - (c)** Samples from NNGP prior. **(d) - (f)** NNGP covariance Φ^L, $L = 20$. **(g) - (i)** NNGP fit (red line) using four training examples (blue dots) sampled from a simple sinusoidal function (green line) with $\sigma_\varepsilon = 0.1$.

Bayesian inference, but with a strong prior for near constant functions with high uncertainty away from the training points.

5 Discussion

We have shown that critical initialisation of noisy ReLU networks corresponds to a choice of optimal kernel parameters in noisy NNGPs and that deviation from these critical parameters leads to poor performance, becoming more severe with depth and the extent of the deviation. In addition, we highlighted the effect of noise on the covariance of a noisy NNGP at criticality, with noise in the limit yielding a fully diagonal covariance, acting as a regulariser on the posterior predictive.

It is interesting to reflect on the connection between deep NNs and GPs in the context of representation learning and noise regularisation. The core assumption in deep learning is that deep NNs learn distributed hierarchical representations useful for modeling the true underlying data generating mechanism, whereas shallow models do not. In these deeper models, noise regularisation is thought to be successful largely because of its influence on representations at different levels of abstraction during the training procedure (Goodfellow et al., 2016). As discussed in previous work (Neal, 1994; Matthews et al., 2018), the kernels associated with NNGPs do not use learned hierarchical representations. Nevertheless these models are still able to perform as well, or sometimes better than, their neural network counterparts (Lee et al., 2018). In the infinite width setting, the success of regularisation from noise injection is unlikely to have the same interpretation as in the finite width setting. We note that in the context of NNGPs, noise regularisation has a stronger connection with controlling the length scale parameter in commonly used kernel functions than regularising through corrupted representations at different levels of abstraction. This connection with the length scale parameter means that noise regularisation in NNGPs may be more accurately interpreted as a useful mechanism to designing priors by controlling the smoothness of the kernel function.

Finally, recent work related to NNGPs, has made it possible to accurately model the learning dynamics of deep neural networks by taking a function space perspective of gradient descent training in the infinite width limit (Jacot et al., 2018; Lee et al., 2019). We envision a similar analysis could be applied to accurately model the learning dynamics of noise regularised deep neural networks.

Stellenbosch University https://scholar.sun.ac.za

References

A. Damianou and N. Lawrence, "Deep Gaussian processes," in *Artificial Intelligence and Statistics*, 2013.

D. Duvenaud, O. Rippel, R. Adams, and Z. Ghahramani, "Avoiding pathologies in very deep networks," in *Artificial Intelligence and Statistics*, 2014.

J. Hensman and N. D. Lawrence, "Nested variational compression in deep Gaussian processes," *arXiv preprint arXiv:1412.1370*, 2014.

Z. Dai, A. Damianou, J. González, and N. Lawrence, "Variational auto-encoded deep Gaussian processes," *arXiv preprint arXiv:1511.06455*, 2015.

T. Bui, D. Hernández-Lobato, J. Hernández-Lobato, Y. Li, and R. Turner, "Deep Gaussian processes for regression using approximate expectation propagation," in *International Conference on Machine Learning*, 2016.

H. Salimbeni and M. Deisenroth, "Doubly stochastic variational inference for deep Gaussian processes," in *Advances in Neural Information Processing Systems*, 2017.

A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep kernel learning," in *Artificial Intelligence and Statistics*, 2016.

A. G. Wilson, Z. Hu, R. R. Salakhutdinov, and E. P. Xing, "Stochastic variational deep kernel learning," in *Advances in Neural Information Processing Systems*, 2016.

M. Al-Shedivat, A. G. Wilson, Y. Saatchi, Z. Hu, and E. P. Xing, "Learning scalable deep kernels with recurrent structure," *Journal of Machine Learning Research*, 2016.

Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *International Conference on Machine Learning*, 2016.

R. M. Neal, "Priors for infinite networks," *Technical report no. crg-tr-94-1, University of Toronto*, 1994.

C. K. Williams, "Computing with infinite networks," in *Advances in Neural Information Processing Systems*, 1997.

N. Le Roux and Y. Bengio, "Continuous neural networks," in *Artificial Intelligence and Statistics*, 2007.

T. Hazan and T. Jaakkola, "Steps toward deep kernel methods from infinite neural networks," *arXiv preprint arXiv:1508.05133*, 2015.

J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, "Deep neural networks as Gaussian processes," *International Conference on Learning Representations*, 2018.

A. G. Matthews, M. Rowland, J. Hron, R. E. Turner, and Z. Ghahramani, "Gaussian process behaviour in wide deep neural networks," *International Conference on Learning Representations*, 2018.

R. Novak, L. Xiao, J. Lee, Y. Bahri, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein, "Bayesian convolutional neural networks with many channels are Gaussian processes," *International Conference on Learning Representations*, 2019.

B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli, "Exponential expressivity in deep neural networks through transient chaos," in *Advances in Neural Information Processing Systems*, 2016.

S. S. Schoenholz, J. Gilmer, S. Ganguli, and J. Sohl-Dickstein, "Deep information propagation," *International Conference on Learning Representations*, 2017.

J. Pennington, S. Schoenholz, and S. Ganguli, "Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice," in *Advances in Neural Information Processing Systems*, 2017.

G. Yang and S. Schoenholz, "Mean field residual networks: On the edge of chaos," in *Advances in Neural Information Processing Systems*, 2017.

M. Chen, J. Pennington, and S. S. Schoenholz, "Dynamical isometry and a mean field theory of RNNs: Gating enables signal propagation in recurrent neural networks," *International Conference on Machine Learning*, 2018.

L. Xiao, Y. Bahri, J. Sohl-Dickstein, S. S. Schoenholz, and J. Pennington, "Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks," *International Conference on Machine Learning*, 2018.

X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Artificial Intelligence and Statistics*, 2010.

K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.

N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, 2014.

A. Pretorius, E. Van Biljon, S. Kroon, and H. Kamper, "Critical initialisation for deep signal propagation in noisy rectifier neural networks," in *Advances in Neural Information Processing Systems*, 2018.

Stellenbosch University https://scholar.sun.ac.za

C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT Press Cambridge, MA, 2006.

Y. Cho and L. K. Saul, "Kernel methods for deep learning," in *Advances in Neural Information Processing Systems*, 2009.

A. Daniely, R. Frostig, and Y. Singer, "Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity," in *Advances In Neural Information Processing Systems*, 2016.

C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," in *Advances in Neural Information Processing Systems*, 2018.

J. Lee, L. Xiao, S. S. Schoenholz, Y. Bahri, J. Sohl-Dickstein, and J. Pennington, "Wide neural networks of any depth evolve as linear models under gradient descent," *arXiv preprint arXiv:1902.06720*, 2019.

Stellenbosch University https://scholar.sun.ac.za

Appendix

A Additional results

Figure 6 provides additional results using CIFAR-10 instead of MNIST for the experiments presented in Figure 1. The results are similar to those described in the main text for MNIST.

B Kernels with non-zero biases

In our experiments, we noticed that noisy ReLU NNGPs often benefit from small non-zeros biases. Therefore, we consider here the implication of non-zero biases for the evolution of the diagonal terms in the NNGP covariance. Recall that the diagonal (variance) terms of the covariance matrix can be expanded as follows

$$
\begin{aligned}
k^L(\mathbf{x},\mathbf{x}) &= \frac{\sigma_w^2}{2}\left(\frac{\sigma_w^2}{2}k^{L-2}(\mathbf{x},\mathbf{x})\odot\mu_2+\sigma_b^2\right)\odot\mu_2+\sigma_b^2 \\
&\vdots \\
&= \begin{cases} \left(\frac{\sigma_w^2}{2}\right)^L k^0(\mathbf{x},\mathbf{x}) + \sum_{l=0}^{L-1}\left(\frac{\sigma_w^2}{2}\right)^l(\mu_2+\sigma_b^2), \\ \qquad \text{if } \odot \equiv + \text{ (Additive noise)}, \\ \left(\frac{\sigma_w^2\mu_2}{2}\right)^L k^0(\mathbf{x},\mathbf{x}) + \sum_{l=0}^{L-1}\left(\frac{\sigma_w^2\mu_2}{2}\right)^l\sigma_b^2, \\ \qquad \text{if } \odot \equiv \times \text{ (Multiplicative noise)}. \end{cases}
\end{aligned} \tag{10}
$$

We first focus on the multiplicative noise case, at the critical weight variance $\sigma_w^2 = \frac{2}{\mu_2}$. Here, a non-zero bias translates into a second term $(L-1)\sigma_b^2$ in (10), that grows linearly with depth. For small initialised biases in typically deep networks this term will be small. For example, a 20-layer deep neural network with $\sigma_b^2 = 0.05$, will translate into an NNGP covariance matrix with diagonal covariance terms given by $k^0(\mathbf{x},\mathbf{x}) + 1$. Therefore, at depth, the linear growth in a non-zero σ_b^2, is far less severe than the exponential growth/decay from an incorrect setting of σ_w^2. In the additive noise case with $\sigma_w^2 = 2$, the situation is similar, but with an added linear growth in noise. Unfortunately, it is less straightforward to analyse the effects of non-zero biases on the off-diagonal covariance terms.

Stellenbosch University https://scholar.sun.ac.za

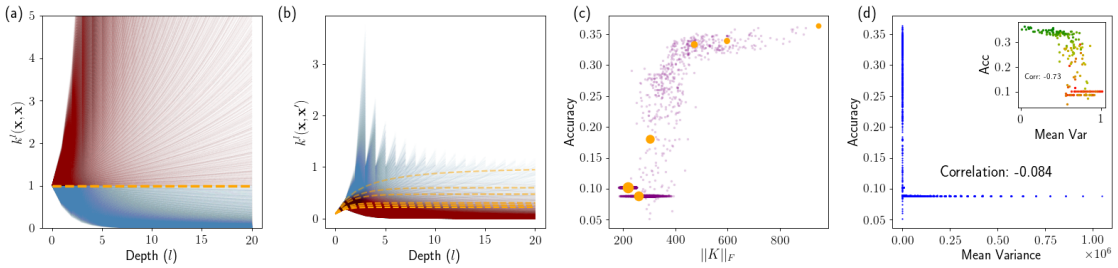


Figure 6: *Effects of noise induced regularisation on NNGPs in Figure 1 for CIFAR-10.* **(a)** Depth evolution of $k^l(\mathbf{x}, \mathbf{x})$ for the different kernel parameters $\sigma_w^2 = 2/\mu_2$ (dashed orange lines), $\sigma_w^2 > 2/\mu_2$ (solid red lines) and $\sigma_w^2 < 2/\mu_2$ (solid blue lines), with $\sigma_b^2 = 0$ in all experiments. **(b)** Depth evolution of $k^l(\mathbf{x}, \mathbf{x}')$ **(c)** Relationship between accuracy and covariance norm. Orange points correspond to critical kernel parameters with larger sizes indicating more noise. **(d)** Quality of uncertainty estimates as measured by the correlation of the mean posterior predictive variance with test accuracy. Main plot contains all points, whereas the inset only contains points close to criticality (green to red showing an increase in noise).

Stellenbosch University https://scholar.sun.ac.za

5.4 Discussion

In **this chapter**, we connected noise regularised deep neural networks with Gaussian processes (GPs) and related the initialisation of the network to the kernel parameters of the GP. Specifically, we found that the critical initialisation derived in Chapter 3 corresponds to the optimal GP kernel parameters. Furthermore, we showed how the covariance of the resulting GP reacts to noise injection, with more noise leading to simpler functions away from the training points and greater uncertainty.

We note that noise regularised neural networks as GPs (noisy NNGPs) are likely to be limited in their applicability. Furthermore, NNGPs operate in what has been referred to as the "lazy learning" regime and do not correspond to the type of learning associated with obtaining rich internal representations (Chizat and Bach, 2018). Some success has been achieved in modeling the learning dynamics of deep neural networks using this point of view (Lee *et al.*, 2019; Arora *et al.*, 2019), which seems promising. However, it is still not entirely clear if lazy learning is an accurate description of how learning actually works in wide deep neural networks (Geiger *et al.*, 2019).

Our work in this chapter rather explores how the connection between neural networks and GPs can enhance our understanding of how noise acts a regulariser. From a practical perspective, *training* a deep neural network to perform exact Bayesian inference is largely intractable. Instead, we must resort to approximation methods and as a result can only consider *approximate* Bayesian inference when it comes to deep learning.

In the **next chapter**, we trade precision for flexibility. We consider training deep Bayesian neural networks (BNNs) for approximate Bayesian inference. Although these networks are often more adaptable to a variety of complex high-dimensional tasks, they are also far more difficult to train. Part of the difficulty in training relates to high variances during signal propagation. Therefore, it is conceivable that by stabilising the propagating signal at each iteration during training, we might be able to alleviate some of the issues plaguing BNN training. In the following chapter, we use some of the ideas from this chapter, as well as Chapter 3, to make training BNNs more robust.

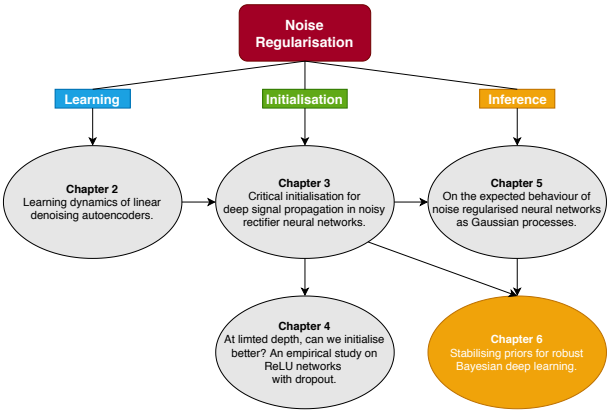
Stellenbosch University https://scholar.sun.ac.za

Chapter 6

Stabilising priors for robust Bayesian deep learning

Paper
McGregor, F., Pretorius, A., du Preez, J., & Kroon, S. Stabilising priors for robust Bayesian deep learning. *(NeurIPS workshop on Bayesian Deep Learning, 2019).*

Code
McGregor, F. (2019). Code for: Stabilising priors for robust Bayesian deep learning. Available at: https://github.com/felixmcgregor/Bayesian-Neural-Networks-with-self-stabilising-priors



6.1 Introduction

In Chapter 5, we examined noise regularised neural networks as Gaussian processes (GPs). These neural network GPs perform exact Bayesian inference. However, inference in GPs can be very expensive, since it typically requires inverting a matrix that grows with the size of the training set. Compared to modern trainable deep neural networks, neural network GPs are more limited in their applicability.

A more direct approach to Bayesian inference in deep neural networks is to directly specify a prior distribution $p_\alpha(W)$, with parameters α, over the weights of the network and use Bayes' rule to find the posterior distribution over the weights. Unfortunately, this approach is not so simple. Applying Bayes' rule for deep networks with an extremely large

Stellenbosch University https://scholar.sun.ac.za

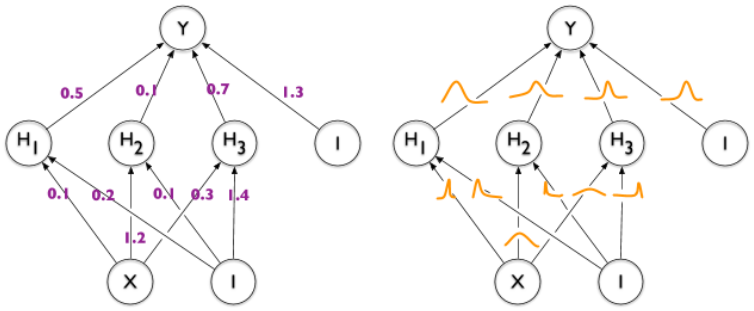


Figure 6.1: *Bayesian neural networks* (Blundell *et al.*, 2015). **Left**: Standard neural network where the parameters have deterministic values (shown in purple). **Right**: Bayesian neural network where the parameters each have their own distribution (shown in orange).

number of weights is computationally intractable. Instead, Bayesian deep neural networks (BNNs) can only perform *approximate* Bayesian inference and rely on *variational inference* (VI) techniques (Blei *et al.*, 2017) to work.

Even when using VI, BNNs remain difficult to train. Stochastic optimisation methods for these models tend to exhibit high variance and BNNs can be very sensitive to small changes in hyperparameters, architecture and the choice of prior.

In this chapter, we develop a new technique for training BNNs. We introduce a *self-stabilising prior* distribution and reformulate the *evidence lower bound* (ELBO) to obtain Monte Carlo estimates by sampling not from the variational distribution $q_\varphi(W)$ (with distributional parameters φ), but rather from a prior-dependent product distribution $\tilde{q}_{\{\alpha,\varphi\}}(W) = p_\alpha(W)q_\varphi(W)$. The purpose of incorporating the prior here is to influence the signal propagation dynamics of the network. The prior adjusts its distributional parameters during training to ensure stability of the propagating signal. We see this approach as a heuristic similar in nature to an iterated application of *empirical Bayes* (EB) for setting prior hyperparameters, as is common in BNN training (e.g. both Graves (2011) and Titsias and Lázaro-Gredilla (2014) propose closed form updates for prior hyperparameters). While EB chooses prior hyperparameters that optimise the likelihood, our approach chooses prior hyperparameters that stabilise signal propagation behaviour in the network. This allows us to effectively train deeper BNNs than otherwise possible as well as improve performance in the previously trainable regime. We further note that this is the first application of signal propagation theory for neural networks outside of initialisation strategies that we are aware of.

6.2 Background

Bayesian neural networks (BNNs) attempt to model the full distribution of the data by approximating distributions over the parameters of the network during training. This is in contrast to standard neural networks that instead use deterministic value updates to model only the mean of the data distribution (see Figure 6.1).

It is typical in BNN training that only the weights of the network have their own distributions and that the biases are treated as being deterministic. During optimisation, a BNN computes gradients with respect to the distributional parameters associated with the weights, instead of the weights themselves, and updates these parameters to better approximate a posterior distribution over each weight. This is achieved using variational inference.

Stellenbosch University https://scholar.sun.ac.za

Variational inference (VI) approaches estimation of the posterior distribution $p(W|\mathcal{D})$, given data $\mathcal{D}$, as an optimisation problem. The goal in VI is to optimise a set of parameters φ associated with an *approximating posterior distribution* $q_\varphi(W)$ to obtain an approximation of the true posterior. Specifically, VI attempts to minimise the Kullback-Leibler divergence

$$\mathrm{KL}(q_\varphi(W)||p(W|\mathcal{D})) = -\int q_\varphi(W) \log \frac{p(W|\mathcal{D})}{q_\varphi(W)} dW \tag{6.2.1}$$

between $q_\varphi(W)$ and the true posterior $p(W|\mathcal{D})$ by maximising the *evidence lower bound* (ELBO)

$$\mathcal{L}_q = L_\mathcal{D}(\varphi, \theta) - \mathrm{KL}(q_\varphi(W)||p_\alpha(W)), \tag{6.2.2}$$

where

$$L_\mathcal{D}(\varphi, \theta) = \sum_{i=1}^{N} \mathbb{E}_{q_\varphi(W)}[\log p(\mathbf{y}_i|\mathbf{x}_i, \theta)]. \tag{6.2.3}$$

We can estimate the expectations in (6.2.2) efficiently using Monte Carlo sampling. Specifically, we reparameterise $q_\varphi(W)$ to be a deterministic function of its parameters transformed by a random variable from a reference distribution. For example, instead of sampling directly from a Gaussian distribution for a scalar weight $w \sim \mathcal{N}(\mu_w, \sigma_w^2)$, where $\varphi = \{\mu_w, \sigma_w^2\}$, we reparameterise $w = \mu_w + \sigma_w \epsilon$ and sample $\epsilon \sim \mathcal{N}(0, 1)$. We can now update the distributional parameters φ using standard gradient based optimisation methods. This is often called the *reparameterisation trick* in Monte Carlo variational inference (MCVI) (Kingma and Welling, 2013; Rezende *et al.*, 2014).

In Chapter 5, we provided general background on Bayesian inference with a fixed prior distribution. *Empirical Bayes* (EB) can be seen as an approximation to full Bayesian inference. Instead of fixing the choice of prior, or placing additional distributions over the parameters of the prior as in hierachical Bayesian inference, EB sets the distributional parameters of the prior using the maximum likelihood principle with the training data. This approach does not strictly align with pure Bayesian modeling philosophy. However, the motivation is that if the Bayesian approach can not be followed in full (i.e. putting priors over priors *ad infinitum*), then in the absence of domain expertise it can sometimes make sense to set the prior parameters to their most likely values by using maximum likelihood.

6.3 Contribution statement

Felix McGregor and I came up with the idea for the this paper through several discussions. My role in this paper was more of a supervisory nature. However, I did contribute substantially to the development of the work and the design of the experiments. I specifically derived the results pertaining to the stabilising prior in Section 3.2, while Felix derived the reparameterisation of the ELBO and linked the two components to develop the technique. Felix ran all the experiments. I wrote large sections of the paper, with Felix being primarily responsible for the related work and experiments section. Prof. Johan du Preez gave general feedback and suggestions throughout the project. Dr Steve Kroon gave technical feedback on a draft version of the paper.

Stellenbosch University https://scholar.sun.ac.za

Stabilising priors for robust Bayesian deep learning

Felix McGregor*
Department of E&E Engineering
Stellenbosch University

Arnu Pretorius
Computer Science Division
Stellenbosch University

Johan du Preez
Department of E&E Engineering
Stellenbosch University

Steve Kroon
Computer Science Division
Stellenbosch University

Abstract

Bayesian neural networks (BNNs) have developed into useful tools for probabilistic modelling due to recent advances in variational inference enabling large scale BNNs. However, BNNs remain brittle and hard to train, especially: (1) when using deep architectures consisting of many hidden layers and (2) in situations with large weight variances. We use signal propagation theory to quantify these challenges and propose *self-stabilising priors*. This is achieved by a reformulation of the ELBO to allow the prior to influence network signal propagation. Then, we develop a stabilising prior, where the distributional parameters of the prior are adjusted before each forward pass to ensure stability of the propagating signal. This stabilised signal propagation leads to improved convergence and robustness making it possible to train deeper networks and in more noisy settings.

1 Introduction

Bayesian neural networks (BNNs) offer a way of combining uncertainty estimation with the expressive power of deep learning. Advances in BNNs (Graves, 2011; Blundell et al., 2015; Kingma and Welling, 2013; Rezende et al., 2014) have made it possible to scale these models but not yet to the level of success of modern deep learning. The reason in part, is that BNNs are difficult to train. Stochastic optimisation methods for these models tend to exhibit high variance and BNNs can be very sensitive to small changes in hyperparameters, architecture and the choice of prior. This work builds on Pretorius et al. (2018), which extended the analysis of signal propagation for deterministic ReLU networks to ReLU networks with stochastic regularisation. In light of recent links between stochastic regularisation techniques and variational inference (Gal and Ghahramani, 2016; Kingma et al., 2015), we relate the initialisation techniques derived in Pretorius et al. (2018), to an iteratively updating prior to stabilise the flow of information through ReLU BNNs throughout training.

Signal propagation analysis of BNNs in the infinite width limit leads us to propose *self-stabilising priors* for robust training. We design an iterative prior with distributional parameters derived to preserve the variance of signals propagating forward through the network. This is a heuristic similar in nature to an iterated application of Empirical Bayes (EB) for setting prior hyperparameters, as is common in BNN training (Graves (2011) and Titsias and Lázaro-Gredilla (2014) propose closed form updates for prior hyperparameters). While EB chooses hyperparameters that optimize the likelihood, our approach chooses prior hyperparameters for each forward pass that attempt to optimize signal propagation behaviour in the network. In order for this prior to influence the signal propagation dynamics, we reformulate the *evidence lower bound* (ELBO) objective to allow this prior to exercise its stabilising effect during the forward pass.

*Correspondence: flx.mcgregor@gmail.com

4th workshop on Bayesian Deep Learning (NeurIPS 2019), Vancouver, Canada.

Stellenbosch University https://scholar.sun.ac.za

2 Self-stabilising priors

The prior $p_\alpha(W)$ usually impacts the ELBO through an additive KL term, only affecting the weights after the forward pass, having no effect on the signal propagation dynamics of the network. We instead propose priors that also exert their influence *during* the forward pass, so as to promote stable signal propagation, and improve robustness in deep BNNs. We reformulate the ELBO in terms of a combination of the prior and approximating posterior as $\tilde{q}_{\{\alpha,\phi\}}(W) = p_\alpha(W)q_\phi(W)/Z$. This formulation allows us to estimate an expectation using a Monte Carlo estimator with samples drawn from $\tilde{q}_{\{\alpha,\phi\}}(W)$ instead of $q_\phi(W)$. This ensures that the sampled weights of the network are being influenced by the current prior $p_\alpha(W)$ during the forward pass. From Appendix B we define the ELBO as

$$\mathcal{L}_{\tilde{q}} \coloneqq \mathbb{E}_{p(\epsilon)}\left[\log p(\mathbf{y}|\mathbf{x}, \mathbf{b}, W = \xi(\epsilon, \alpha, \phi))\right] - \mathrm{KL}(\tilde{q}_{\{\alpha,\phi\}}(W)||p_\alpha(W)), \tag{1}$$

where $\epsilon \sim \mathcal{N}(0, I)$. The prior is adjusted after every gradient update to adapt to the updated posterior $q_\phi(W)$. These prior parameters are optimal in the sense that together with the reformulated ELBO in (1), the sampled weights from $\tilde{q}_{\{\alpha,\phi\}}(W)$ have a stabilising effect on the signal propagation dynamics of a Bayesian deep neural network.

To quantify the signal propagation dynamics we make use of the *mean field* assumption (Saxe et al., 2014; Poole et al., 2016) which allows for the components of the pre-activation vectors $\mathbf{h}^l$ to be treated as independent Gaussian random variables. Then, according to the central limit theorem, these pre-activations are Gaussian distributed. We consider independent Gaussian priors $p_\alpha(w^l_{i,j}) = \mathcal{N}(\mu^l_{p_{ij}}, (\sigma^l_{p_{i,j}})^2)$ and posteriors $q_\phi(w_{i,j}) = \mathcal{N}(\mu^l_{q_{ij}}, (\sigma^l_{q_{i,j}})^2)$. Focusing on ReLU activations, i.e. $g(a) = \max(0, a)$, we consider a single scalar hidden unit pre-activation h^l_j at an arbitrary layer l of the network. In Appendix A, we show that in expectation over the weights and biases, under the assumption of zero mean inputs at each layer (true at initialisation but a somewhat unrealistic assumption during training further discussed in Appendix A), the variance of h^l_j, governing signal propagation in the forward pass is given by

$$\mathrm{Var}[h^l_j] = \left[\left(1 - \frac{1}{\pi}\right)(\mu^l_{\tilde{q}_j})^2 + (\sigma^l_{\tilde{q}_j})^2\right]\frac{\mathrm{Var}[h^{l-1}_{j'}]}{2}. \tag{2}$$

where $j' \in \{1, ..., D_{l-1}\}$. Then, to stabilise the signal, we want to find prior parameters $\alpha = \{\mu^l_{p_{ij}}, \sigma^l_{p_{i,j}}\}$ that can preserve the variance during the forward pass. Specifically, we want α to ensure $\mathrm{Var}[h^l_j] = \mathrm{Var}[h^{l-1}_{j'}]$ in (2). To achieve this, we require $\mu^l_{p_{ij}} = \mu^l_{q_{ij}}$ giving the stabilising prior parameters α as

$$\mu^l_{p_{ij}} = \mu^l_{q_{ij}}, \qquad \sigma^l_{p_j} = \sqrt{\frac{(\sigma^l_{\tilde{q}_j})^2\gamma}{(\sigma^l_{\tilde{q}_j})^2 - \gamma}}, \tag{3}$$

where $\gamma = |2 - (1 - 1/\pi)(\mu^l_{q_j})^2|$ and we take the absolute value to ensure positive variances. Note that we can apply the result in (2) recursively for all layers $l = 1, ..., L$, with base case $\mathrm{Var}[x^0] = \frac{1}{D_0}\mathbf{x}^0 \cdot \mathbf{x}^0$. Therefore, sampling the weights $W \sim \tilde{q}_{\{\alpha,\phi\}}(W)$ at each forward pass, while setting the prior $p_\alpha(w) = \mathcal{N}(\mu^l_q, |(\sigma^l_q)^2\gamma/((\sigma^l_q)^2 - \gamma)|/D_{l-1})$, promoting stable signal propagation.

The effect of the stabilising prior is that the larger the mean and variance of the incoming weights, the more likely it is to destroy the signal, whereas if the second moment is small, it is not likely to add noise to the signal. The prior encourages the weight distribution to sample closer to its means when the second moment of the distribution is large.

3 Experiments

Limits of trainability. We restrict ourselves to BNNs with fully connected layers with a specified number of hidden layers all of constant width. [2]. We investigate performance at extremities by training a series of networks with varying depths and initial variances. We compare a series of networks with our proposed stabilising prior incorporated on the forward pass with a standard non-conjugate Gaussian prior Titsias and Lázaro-Gredilla (2014) which we report in Figures 1 (a) and (b). We observe our stabilising prior makes it possible to train deeper BNNs and in more noisy conditions.

[2]Code available `https://git.io/JeLkH`

Stellenbosch University https://scholar.sun.ac.za

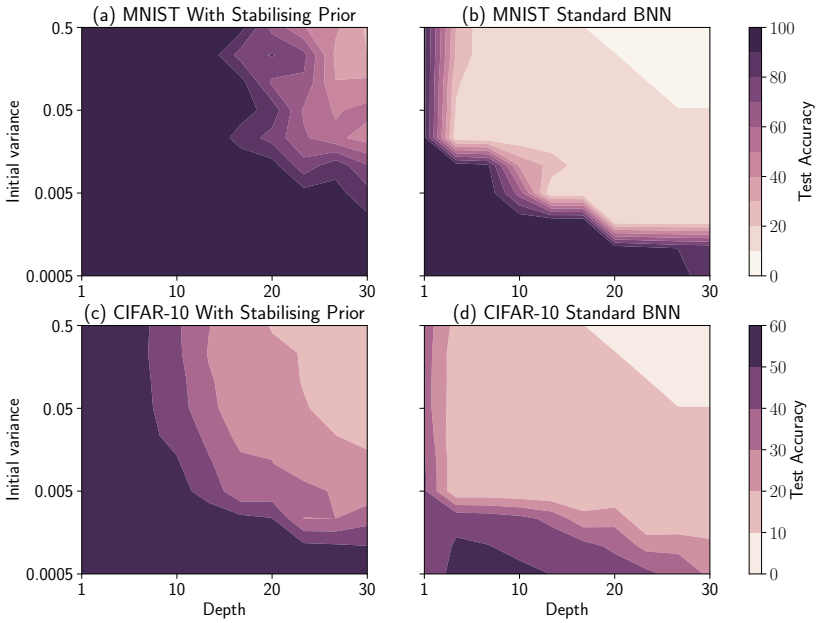


Figure 1: MNIST and CIFAR-10 large scale experiments. Classification accuracy grid of ReLU networks trained with varying depths and initial variance conditions trained for 50 epochs.

Accelerated training. In general, we also observe that our prior improves convergence as demonstrated in Figure 2. We compare with Empirical Bayes (EB) as in Titsias and Lázaro-Gredilla (2014) which uses the gradient to find optimal hyperparameters for the prior. We further compare these priors with a Gaussian prior and report their results for both the reparametrisation trick (RT) Kingma and Welling (2013) and local reprametrisation trick (LRT) Kingma et al. (2015).

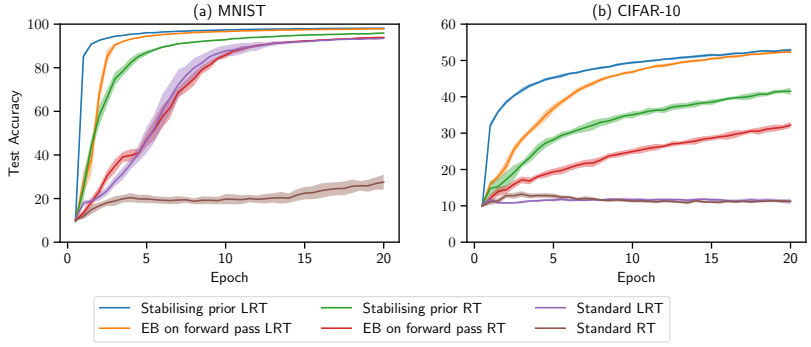


Figure 2: Progression of test accuracy for various networks through training averaged over 10 runs for a 5 layer deep 512 wide network with an initial variance of 0.001.

4 Discussion

We used signal propagation theory to derive priors for BNNs that promote stable signal propagation. The prior incorporates knowledge of model architecture and activation function derived from how signals propagate in the network in the infinite width limit. We showed that these priors, when their effect is exerted in the forward pass, makes it possible to train deeper networks and in more noisy conditions. This alleviates the need to tune hyper-parameters and extends BNNs to deeper architectures. We also observe that stable signal propagation accelerates training, which we attribute to cleaner signals and gradients being propagated through the network and more efficient expectations.

Stellenbosch University https://scholar.sun.ac.za

Acknowledgements

We would like to thank the Technology and Human Resource for Industry Programme (THRIP) as well as Centre for Artificial Intelligence Research (CAIR) for financial support. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

A. Graves, "Practical variational inference for neural networks," in *Advances in neural information processing systems*, 2011, pp. 2348–2356.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," *arXiv preprint arXiv:1505.05424*, 2015.

D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," *arXiv preprint arXiv:1401.4082*, 2014.

A. Pretorius, E. Van Biljon, S. Kroon, and H. Kamper, "Critical initialisation for deep signal propagation in noisy rectifier neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 5722–5731.

Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 2016, pp. 1050–1059.

D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Advances in Neural Information Processing Systems*, 2015, pp. 2575–2583.

M. Titsias and M. Lázaro-Gredilla, "Doubly stochastic variational bayes for non-conjugate inference," in *International conference on machine learning*, 2014, pp. 1971–1979.

A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *Proceedings of the International Conference on Learning Representations*, 2014.

B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli, "Exponential expressivity in deep neural networks through transient chaos," in *Advances in Neural Information Processing Systems*, 2016, pp. 3360–3368.

A. Wu, S. Nowozin, E. Meeds, R. E. Turner, J. M. Hernandez-Lobato, and A. L. Gaunt, "Deterministic variational inference for robust bayesian neural networks," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=B1l08oAct7

S. S. Schoenholz, J. Gilmer, S. Ganguli, and J. Sohl-Dickstein, "Deep information propagation," *Proceedings of the International Conference on Learning Representations*, 2017.

B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems*, 2017, pp. 6402–6413.

Stellenbosch University https://scholar.sun.ac.za

Appendices

A Signal propagation in BNNs

We consider the quantity $\mathbb{E}[(h_j^l)^2]$ during a forward pass. This quantity is the second moment of a single hidden unit h_j^l in layer l, consisting of D_{l-1} incoming connections, where the expectation taken is over the network parameters and is given by

$$\begin{aligned}\mathbb{E}[(h_j^l)^2] &= \mathbb{E}\left[\left(\sum_{i=1}^{D_{l-1}} w_{ij}^l x_i^l + b_j^l\right)^2\right] \\ &= \sum_{i=1}^{D_{l-1}} \mathbb{E}[(w_{ij}^l)^2]\mathbb{E}[(x_i^l)^2] + \mathbb{E}[(b_j^l)^2]. \end{aligned} \tag{4}$$

We define

$$\sigma_{\tilde{q}_j}^l = \sum_{i=1}^{D_{l-1}} (\sigma_{ij}^l)^2 \qquad \text{and} \qquad \mu_{\tilde{q}_j}^l = \sum_{i=1}^{D_{l-1}} \mu_{ij}^l \tag{5}$$

to use as the statistics to describe $\sum_{i=1}^{D_{l-1}} w_{ij}$. Note that while it is true that we can write

$$\sum_{i=1}^{D_{l-1}} w_{ij}^l \sim \mathcal{N}\left(\mu_{\tilde{q}_j}^l, (\sigma_{\tilde{q}_j}^l)^2\right) \tag{6}$$

at initialisation, in our analysis of the network at an arbitrary stage of the stage of training the i.i.d. assumption of the central limit theorem (CLT) does not strictly hold. As in Wu et al. (2019), we empirically find that some form of the CLT holds for the hidden units during training. We thus continue to approximate the expectation with a Gaussian according to the CLT.

Next, in designing $\tilde{q}_\beta(W)$ we scale the variance and impose $\mathbb{E}[(w_{ij}^l)^2] = \frac{(\mu_{\tilde{q}}^l)^2 + (\sigma_{\tilde{q}}^l)^2}{D_{l-1}}, \forall i, j$ to ensure that the variance is bounded in the infinite width limit Schoenholz et al. (2017). This also allows the variance propagated forward to be independent of the layer width. We now have

$$\mathbb{E}[(h_j^l)^2] = ((\mu_{\tilde{q}_j}^l)^2 + (\sigma_{\tilde{q}_j}^l)^2)\frac{1}{D_{l-1}} \sum_{i=1}^{D_{l-1}} g(h_i^{l-1})^2 + ((\mu_b^l)^2 + (\sigma_b^l)^2). \tag{7}$$

As $D_{l-1} \to \infty$, h_j^l becomes an infinite sum of i.i.d. random variables and becomes Gaussian distributed according to the CLT. We can thus write

$$\mathbb{E}[(h_j^l)^2] = ((\mu_{\tilde{q}}^l)^2 + (\sigma_{\tilde{q}_j}^l)^2)\mathbb{E}_z[\phi(\tau^{l-1} + \sqrt{\nu^{l-1}}z)^2] + (\mu_b^l)^2 + (\sigma_b^l)^2 \tag{8}$$

where $z \sim \mathcal{N}(0, 1)$, and τ^{l-1} and ν^{l-1} are the incoming signal to layer l's mean and variance respectively. If we use ReLU as activation, i.e. $g(a) = \max(0, a)$, then

$$\begin{aligned}\mathbb{E}[(h_j^l)^2] &= ((\mu_{\tilde{q}}^l)^2 + (\sigma_{\tilde{q}_j}^l)^2)\left\{\int_{-\infty}^{\infty} \Phi(z)\phi(\tau^{l-1} + \sqrt{\nu^{l-1}}z)^2 dz\right\} + (\mu_b^l)^2 + (\sigma_b^l)^2 \\ &= ((\mu_{\tilde{q}}^l)^2 + (\sigma_{\tilde{q}_j}^l)^2)\left\{\int_0^{\infty} \Phi(z)\left((\tau^{l-1})^2 + 2\tau^{l-1}\sqrt{\nu^{l-1}}z + \nu^{l-1}z^2\right) dz\right\} \\ &\qquad + (\mu_b^l)^2 + (\sigma_b^l)^2 \\ &= ((\mu_{\tilde{q}}^l)^2 + (\sigma_{\tilde{q}_j}^l)^2)\left[\frac{(\tau^{l-1})^2}{2} + \frac{2\tau^{l-1}\sqrt{\nu^{l-1}}}{\sqrt{2\pi}} + \frac{\nu^{l-1}}{2}\right] + (\mu_b^l)^2 + (\sigma_b^l)^2 \end{aligned} \tag{9}$$

where $\Phi(z) = \frac{e^{-z^2/2}}{\sqrt{2\pi}}$.

Stellenbosch University https://scholar.sun.ac.za

Similarly, we can show that the relevant statistics governing signal propagation in the forward pass are given by

$$\begin{aligned}\mathbb{E}[h_j^l] &= \mathbb{E}\left[\sum_{j=1}^{D_{l-1}} w_{ij}^l g(h_j^{l-1} + b_i)\right] \\ &= \mu_{\tilde{q}_j}^l \left(\frac{\tau^{l-1}}{2} + \sqrt{\frac{\nu^{l-1}}{2\pi}}\right) + \mu_b^l \end{aligned} \tag{10}$$

and

$$\begin{aligned}\nu_j^l &= \mathbb{E}\left[\left(\sum_{j=1}^{D_{l-1}} w_{ij}^l g(h_j^{l-1})\right)^2\right] - \mathbb{E}\left[\sum_{j=1}^{D_{l-1}} w_{ij}^l g(h_j^{l-1})\right]^2 \\ &= (\mu_{\tilde{q}_j}^l)^2 \left[\frac{(\tau^{l-1})^2}{4} + \tau^{l-1}\sqrt{\frac{\nu^{l-1}}{2\pi}} + \left(1 - \frac{1}{\pi}\right)\frac{\nu^{l-1}}{2}\right] \\ &\quad + (\sigma_{\tilde{q}_j}^l)^2 \left[\frac{(\tau^{l-1})^2}{2} + 2\tau^{l-1}\sqrt{\frac{\nu^{l-1}}{2\pi}} + \frac{\nu^{l-1}}{2}\right] + (\sigma_b^l)^2. \end{aligned} \tag{11}$$

Described above is the signal propagation dynamics in general. With a mean preserving prior we can only control variance by multiplicatively expanding or squeezing $\sigma_{\tilde{q}_j}^l$. We can only design for conditions that set $\tau^{l-1} = 0$ and $\sigma_b = 0$ which is true at initialisation but starts to break down during training. In order to continue we thus implicitly assume that during training: (1) the mean of the summed weights' means across a hidden layer's pre-activation remain mean zero i.e. $\mathbb{E}[(\sum_{j=1}^{D_l} \mu_{q_j}^l)] = 0$; (2) biases are zero (note, it is possible to absorb the biases by augmenting the input at each layer with an additional column of ones, this yields more stable signal propagation. We find that treating biases as deterministic parameters aids in training and outweighs the minor gain in stabilising propagation). This allows us to write the variance ν_j^l as

$$\nu_j^l = \left[\left(1 - \frac{1}{\pi}\right)(\mu_{\tilde{q}_j}^l)^2 + (\sigma_{\tilde{q}_j}^l)^2\right]\frac{\nu^{l-1}}{2}. \tag{12}$$

From here we can design $\tilde{q}_{\{\alpha,\phi\}}(W)$ to preserve variances through the network. We find actually forcing our assumptions and setting parameter means and biases to zero does not train.

The parameters for the joint distribution $\tilde{q}_{\{\alpha,\phi\}}(W)$ are determined by α and ϕ as the product of two Gaussian pdfs as

$$\mu_{\tilde{q}_{ij}}^l = \frac{\mu_{q_{ij}}^l (\sigma_{p_{ij}}^l)^2 + \mu_{p_{ij}}^l (\sigma_{q_{ij}}^l)^2}{(\sigma_{p_j}^l)^2 + (\sigma_{q_{ij}}^l)^2}, \qquad \sigma_{\tilde{q}_{ij}}^l = \sqrt{\frac{(\sigma_{p_j}^l)^2 (\sigma_{q_{ij}}^l)^2}{(\sigma_{p_j}^l)^2 + (\sigma_{q_{ij}}^l)^2}}. \tag{13}$$

To stabilise the signal, we want to find prior parameters $\alpha \in \{\mu_{p_{ij}}^l, (\sigma_{p_j}^l)^2\}$ that can preserve the variance during the forward pass. Specifically, we want α to ensure $\nu_j^l = \nu_j^{l-1}$ or $\text{Var}[h_j^l] = \text{Var}[h_{j'}^{l-1}]$ in (2) where $j' \in \{1, ..., D_{l-1}\}$. To achieve this, we require $\mu_{p_{ij}}^l = \mu_{q_{ij}}^l$. Secondly, we find variance parameters of $(\sigma_{p_j}^l)^2$ using (13), to satisfy the condition $(1 - 1/\pi)(\mu_{\tilde{q}_j}^l)^2 + (\sigma_{\tilde{q}_j}^l)^2 = 2$ found by setting $\nu_j^l = \nu_j^{l-1}$ in 12.

B Reformulating the ELBO

We reformulate the ELBO by lower bounding the log marginal likelihood of the data as follows

$$\begin{aligned}\log p(\mathbf{y}|\mathbf{x}) &= \log \int p(\mathbf{y}|\mathbf{x}, W) p_\alpha(W) dW \\ &= \log \int p(\mathbf{y}|\mathbf{x}, W) p_\alpha(W) \frac{q_\phi(W)}{q_\phi(W)} dW, \end{aligned} \tag{14}$$

Stellenbosch University https://scholar.sun.ac.za

where we combine the prior and approximating posterior as $\tilde{q}_{\{\alpha,\phi\}}(W) = p_\alpha(W)q_\phi(W)/Z$, where Z is a normalisation constant. Then, we can construct a lower bound making use of Jensen's inequality which gives

$$\begin{aligned}\log p(\mathbf{y}|\mathbf{x}) &= \log \int \tilde{q}_{\{\alpha,\phi\}}(W)\frac{p(\mathbf{y}|\mathbf{x},W)}{q_\phi(W)}ZdW \\ &\geq \int \tilde{q}_{\{\alpha,\phi\}}(W)\log\frac{p(\mathbf{y}|\mathbf{x},W)}{q_\phi(W)}ZdW \\ &= \mathbb{E}_{\tilde{q}_{\{\alpha,\phi\}}(W)}\left[\log p(\mathbf{y}|\mathbf{x},W)\right] - \mathbb{E}_{\tilde{q}_{\{\alpha,\phi\}}(W)}\left[\log q_\phi(W)\right] + \log Z. \end{aligned} \tag{15}$$

By reformulating the ELBO in this way, we can estimate the above expectations using a Monte Carlo estimator with samples drawn from $\tilde{q}_{\{\alpha,\phi\}}(W)$ instead of $q_\phi(W)$. This ensures that the sampled weights of the network are being influenced by the current prior $p_\alpha(W)$ during the forward pass. Finally, we ignore the constant term in (15) and replace the cross-entropy term, which is the second term in the equation, with a KL term by introducing the identity ratio $p_\alpha(W)/p_\alpha(W)$ in (14). This gives the following objective when reparametrised using the reparametrisation trick:

$$\mathcal{L}_{\tilde{q}} := \mathbb{E}_{p(\epsilon)}\left[\log p(\mathbf{y}|\mathbf{x},\mathbf{b},W=\xi(\epsilon,\alpha,\phi))\right] - \mathrm{KL}(\tilde{q}_{\{\alpha,\phi\}}(W)||p_\alpha(W)), \tag{16}$$

where $\epsilon \sim \mathcal{N}(0,I)$. The prior $p_\alpha(W)$ usually impacts the ELBO through an additive KL term, only affecting the weights after the forward pass, having no effect on the signal propagation dynamics of the network. We instead propose priors that also exert their influence *during* the forward pass, so as to promote stable signal propagation, and improve robustness in deep BNNs.

C Use of $\tilde{q}(W)$ at test time

We opt to always include the prior during test time: we sample from $\tilde{q}(W)$ instead of $q(W)$, defining a new posterior of interest, because we find that the accuracy of the model progresses faster. Since we require adjustment during the forward pass throughout training to ensure stable signal propagation (it is often necessary for any training to occur), it seems reasonable to include it in the forward pass at test time. The use of $q(W)$ exhibits performance similar to networks with unstable signal propagation. Note that the adjustment to the variational posterior for signal propagation is unlike the use of EB to choose the hyperparameters maximizing the likelihood, in that case the prior should not be factored in and $q(W)$ would be appropriate. We investigate what effect including the prior has on the quality of prediction uncertainty in the experiments in Appendix D.

D Appendix Experiments: Signal Propagation and Quality of Uncertainty

Signal propagation. We examine signals in a ReLU network in Figure 3 to analyse the effect of the prior on the network signal propagation. We monitor the variance dynamics of the same data point throughout training by calculating the empirical variance of the vector of pre-activations at each layer during a forward pass. In Figures 3 (a) and (b) we show a controlled example where we force our assumptions, setting biases and parameter means to zero and see that the prior preserves the signal, whereas in a standard BNN it explodes. Furthermore, we show a typical training scenario in Figures 3 (c) and (d), where we see that our assumptions hold in the early stages of training and start to break down later in training, yielding less stable signal propagation.

Quality of uncertainty. Finally, we turn to the issue of what effect this prior has on uncertainty and calibration. We measure calibration with the Brier score and, similar to Lakshminarayanan et al. (2017), the accuracy of predictions above 50% and 90% confidence to see whether our models tend towards overconfidence. We monitor the progression of these metrics of models with different priors through 100 epochs reported in Figure 4. As with any iteratively updating prior, we expect that it may adapt to the dataset and overfit, as is shown to be true of our stabilising prior and EB in Figure 4. As an answer to this we explore combining a regularising and stabilising prior which trains faster and results in a well calibrated model with better Brier scores than any solitary prior.

Stellenbosch University https://scholar.sun.ac.za

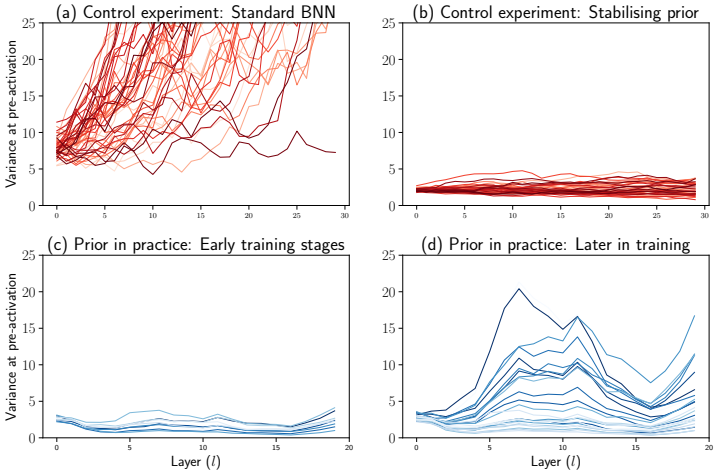


Figure 3: Signal propagation dynamics of the same signal propagated through different networks. We track the variance of the same data point throughout training by calculating the empirical variance of the vector at the pre-activation at each layer. In a controlled setting we can achieve perfectly stable signal propagation. In practice our assumptions hold for the early stages of training.

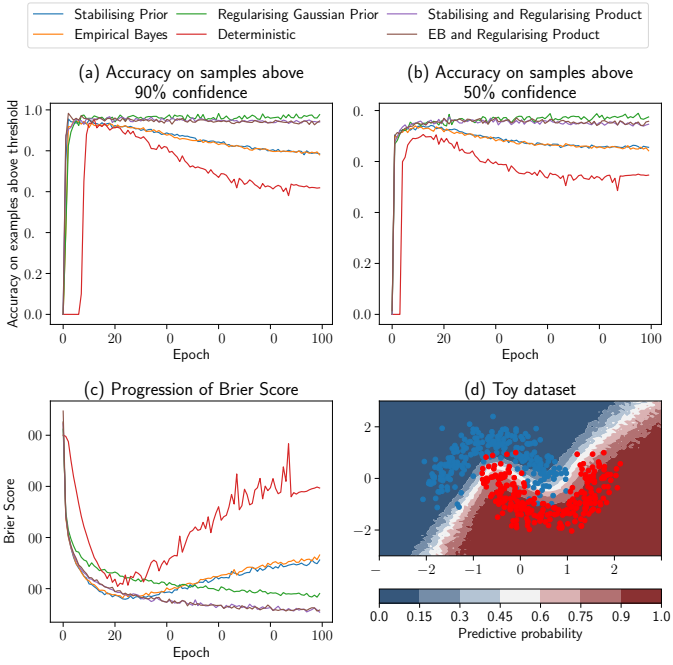


Figure 4: Uncertainty and calibration experiments on CIFAR-10. Iteratively updating priors overfit, however, we can combine regularising and optimal priors to maintain calibrated confidence and better Brier scores. In (d) we also see we are able to get reasonable uncertainty estimates with a deep neural network on a toy dataset.

Stellenbosch University https://scholar.sun.ac.za

6.4 Discussion

In this chapter, we saw how we could use some of the ideas from the previous chapters to come up with a method that improved the training of deep Bayesian neural networks (BNNs). By introducing a stabilising prior into the reformalised ELBO objective that adapts its distributional parameters to assist in stable signal propagation at every iteration, we achieve improved robustness and performance in deep BNNs.

In the **next and final chapter**, we conclude with a summary of our contributions as well as give suggestions of potentially fruitful directions for future work.

Stellenbosch University https://scholar.sun.ac.za

Chapter 7

Conclusions

7.1 Summary

Regularisation is central to deep learning. However, modern regularisation techniques are often not developed in a principled manner and serve more as useful heuristic tools that have been shown to work well in practice. Therefore, we still poorly understand how and why different regularisation strategies work, and in what ways these strategies affect the different stages of modeling. We further believe that a better understanding of regularisation for deep learning might lead to more principled approaches and improved techniques.

Our aim in this thesis is to contribute towards an improved understanding of regularisation for deep learning. We specifically focused on an effective regularisation strategy that injects noise into a neural network during training, referred to as *noise regularisation*. Our contributions pertain to an improved understanding of noise regularisation in the following stages of the modeling: (1) learning, (2) initialisation and (3) inference.

In **Chapter 2**, we analysed the **learning** dynamics of denoising autoencoders (DAEs). These models inject noise at the input layer to regularise the network during training. We found that input noise in DAEs help the network to focus on learning directions of higher variation in the input, while ignoring smaller variance directions. We compared the dynamics of DAEs with autoencoders that use weight decay regularisation. We showed that noise regularisation can have similar effects to weight decay, but with training dynamics that allow DAEs to learn the main directions of variation with fewer training iterations. We verified that our theoretical predictions approximate learning dynamics on real-world data and qualitatively match observed dynamics in nonlinear DAEs. Finally, we noted that the learning trajectories for these noise regularised models strongly depended on initialisation.

Chapter 3 considered general noise regularisation in the input and intermediate layers for fully-connected feedforward neural networks. We investigated how noise interacts with parameter **initialisation** and how this affects the signal propagation dynamics of the network. By better understanding how noise impacts the signal, we were able to derive critical initialisation strategies for neural networks that use ReLU activations. We also found that noise regularisation limits the depth to which ReLU networks can train. This depth was shown to depend on the amount of noise injected into the network during training.

If noise limits depth, can we initialise better? This was the question we considered in **Chapter 4**. Specifically, we investigated the **initialisation** landscape around criticality

Stellenbosch University https://scholar.sun.ac.za

in ReLU neural networks of moderate depth that use dropout regularisation. A large-scale controlled experiment revealed that there is no statistically significant difference in training speed or generalisation between the critical initialisation and a wide range of alternative non-critical initialisations.

Chapter 5 connected deep noise regularised neural networks with Gaussian processes (GPs). For fully-connected feedforward neural networks with ReLU activation, we analysed the expected behaviour of their equivalent GPs. We found that the best performing GPs are those that have kernel parameters corresponding to the initialisation derived in Chapter 3. In addition, we showed how the GP covariance becomes progressively more diagonal as more noise is injected. This gives a stronger prior for simple functions and larger uncertainty in the posterior predictive distribution when performing exact Bayesian **inference**.

In **Chapter 6**, we considered approximate Bayesian **inference** in deep neural networks. We developed a novel approach for training deep Bayesian neural networks (BNNs) using *self-stabilising priors*. The iteratively updated prior stabilises network dynamics during training and leads to improved convergence and robustness. Using this technique, we were able to successfully train deeper BNNs in noisier settings and outperform other state-of-the-art methods.

7.2 Future work

Most of the work in this thesis focused on fully-connected feedforward neural network that use ReLU activations. Therefore, a logical extension would be to consider other architectures and activations functions and study the effects of noise regularisation in these settings. For example, how does noise regularisation such as dropout interact with initialisation in recurrent neural networks? In terms of alternative activation functions, *tanh* was the activation primarily studied in prior work (Poole *et al.*, 2016; Schoenholz *et al.*, 2017). And although we chose ReLU because of its widespread use, recent work seems to suggest that sigmoid-like activations could be more suitable for training very deep networks when coupled with orthogonal initialisation schemes (Pennington *et al.*, 2017). Therefore, understanding the effect of noise regularisation under these design choices could be beneficial.

A more specific line of possible future work follows the development of the *Neural Tangent Kernel* (NTK) in deep neural networks (Jacot *et al.*, 2018). Building on the work by Lee *et al.* (2018) connecting neural networks with Gaussian processes, Jacot *et al.* (2018) showed that by taking a function space perspective of gradient descent training in the infinite width limit, the training of a neural network can be described by the NTK. The NTK is random at initialization and varies during training. However, in the limit of infinite width it converges to an explicit limiting kernel which stays constant during training. This has made it possible to accurately model the learning dynamics of deep neural networks and describe their generalisation properties (Lee *et al.*, 2019).

The NTK opens up many new possibilities for future analysis regarding the effects of noise regularisation in deep neural network. For example, it is conceivable that by using the NTK, we could study the effect of initialisation on learning and generalisation and how this interacts with noise regularisation. Other possibilities are developing new noise models and directly testing their effects using the NTK. We could cheaply develop "large scale" experiments by running many different configurations only in terms of simulated

Stellenbosch University https://scholar.sun.ac.za

dynamics. We note that this approach can be extended to many different architectures and neural network designs (Yang, 2019).

Finally, the motivation for noise injection can be rooted in biology. For example, noisy neural activity in the human brain has been shown to be critical for learning (McDonnell and Ward, 2011; Engel *et al.*, 2015). An interesting direction for future work could therefore be to study the relationships between neural network models of noisy learning and neuroscientific models of biological noisy learning that happens in the human brain.

Stellenbosch University https://scholar.sun.ac.za

List of references

Arora, S., Du, S.S., Hu, W., Li, Z., Salakhutdinov, R. and Wang, R. (2019). On exact computation with an infinitely wide neural net. *arXiv preprint arXiv:1904.11955*.

Bellman, R.E. (1961). *Adaptive control processes: a guided tour*. Princeton University Press.

Bengio, Y., Lamblin, P., Popovici, D. and Larochelle, H. (2007). Greedy layer-wise training of deep networks. In: *Advances in Neural Information Processing Systems*, pp. 153–160.

Bishop, C.M. (2006). *Pattern recognition and machine learning*. springer.

Blei, D.M., Kucukelbir, A. and McAuliffe, J.D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877.

Blundell, C., Cornebise, J., Kavukcuoglu, K. and Wierstra, D. (2015). Weight uncertainty in neural networks. *International Conference on Machine Learning*.

Bourlard, H. and Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, vol. 59, no. 4-5, pp. 291–294.

Brea, J., Simsek, B., Illing, B. and Gerstner, W. (2019). Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape. *arXiv preprint arXiv:1907.02911*.

Chizat, L. and Bach, F. (2018). A note on lazy training in supervised differentiable programming. *arXiv preprint arXiv:1812.07956*.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Draxler, F., Veschgini, K., Salmhofer, M. and Hamprecht, F.A. (2018). Essentially no barriers in neural network energy landscape. *Internation Conference on Machine Learning*.

Engel, T.A., Chaisangmongkon, W., Freedman, D.J. and Wang, X.-J. (2015). Choice-correlated activity fluctuations underlie learning of neuronal category representation. *Nature Communications*, vol. 6, p. 6454.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P. and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, vol. 11, pp. 625–660.

Erhan, D., Szegedy, C., Toshev, A. and Anguelov, D. (2014). Scalable object detection using deep neural networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2147–2154.

Geiger, M., Spigler, S., Jacot, A. and Wyart, M. (2019). Disentangling feature and lazy learning in deep neural networks: an empirical study. *arXiv preprint arXiv:1906.08034*.

Stellenbosch University https://scholar.sun.ac.za

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In: *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 249–256.

Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep learning*. MIT press.

Graves, A. (2011). Practical variational inference for neural networks. In: *Advances in Neural Information Processing Systems*.

Graves, A. (2013). Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Graves, A. and Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In: *International Conference on Machine Learning*, pp. 1764–1772.

Hastie, T., Tibshirani, R. and Friedman, J. (2009). *The Elements of Statistical Learning*. Springer, 2nd edition.

He, K., Zhang, X., Ren, S. and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1026–1034.

Hendrycks, D. and Gimpel, K. (2016). Adjusting for dropout variance in batch normalization and weight initialization. *arXiv preprint arXiv:1607.02488*.

Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N. and Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97.

Hinton, G.E. and Zemel, R.S. (1994). Autoencoders, minimum description length and Helmholtz free energy. In: *Advances in Neural Information Processing Systems*, pp. 3–10.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, vol. 9, no. 8, pp. 1735–1780.

Jacot, A., Gabriel, F. and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. In: *Advances in Neural Information Processing Systems*, pp. 8580–8589.

Kamper, H., Elsner, M., Jansen, A. and Goldwater, S. (2015). Unsupervised neural network based feature extraction using weak top-down constraints. In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5818–5822. IEEE.

Kingma, D.P. and Welling, M. (2013). Auto-encoding variational Bayes. *Internation Conference on Learning Representations*.

Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012). ImageNet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems*, pp. 1097–1105.

Krogh, A. and Hertz, J.A. (1992). A simple weight decay can improve generalization. In: *Advances in Neural Information Processing Systems*, pp. 950–957.

Laurent, T. and von Brecht, J. (2017). Deep linear neural networks with arbitrary loss: All local minima are global. *arXiv preprint arXiv:1712.01473*.

Stellenbosch University https://scholar.sun.ac.za

LeCun, Y. (1987). Modeles connexionnistes de l'apprentissage (connectionist learning models). *Ph.D. thesis, Universite de Paris VI*.

LeCun, Y. (1989). Generalization and network design strategies. In: *Connectionism in perspective*, vol. 19.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S.S., Pennington, J. and Sohl-Dickstein, J. (2018). Deep neural networks as Gaussian processes. *International Conference on Learning Representations*.

Lee, J., Xiao, L., Schoenholz, S.S., Bahri, Y., Sohl-Dickstein, J. and Pennington, J. (2019). Wide neural networks of any depth evolve as linear models under gradient descent. *arXiv preprint arXiv:1902.06720*.

Matthews, A.G.d.G., Rowland, M., Hron, J., Turner, R.E. and Ghahramani, Z. (2018). Gaussian process behaviour in wide deep neural networks. *International Conference on Learning Representations*.

McDonnell, M.D. and Ward, L.M. (2011). The benefits of noise in neural systems: bridging theory and experiment. *Nature Reviews Neuroscience*, vol. 12, no. 7, p. 415.

Neal, R.M. (1994). Priors for infinite networks.

Novak, R., Bahri, Y., Abolafia, D.A., Pennington, J. and Sohl-Dickstein, J. (2018). Sensitivity and generalization in neural networks: an empirical study. *International Conference on Learning Representations*.

Pennington, J., Schoenholz, S. and Ganguli, S. (2017). Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In: *Advances in Neural Information Processing Systems*, pp. 4788–4798.

Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J. and Ganguli, S. (2016). Exponential expressivity in deep neural networks through transient chaos. In: *Advances in Neural Information Processing Systems*, pp. 3360–3368.

Räsänen, O. (2012). Computational modeling of phonetic and lexical learning in early language acquisition: Existing models and future directions. *Speech Communication*, vol. 54, no. 9, pp. 975–997.

Räsänen, O. and Rasilo, H. (2015). A joint model of word segmentation and meaning acquisition through cross-situational learning. *Psychological review*, vol. 122, no. 4, p. 792.

Rezende, D.J., Mohamed, S. and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In: *International Conference on Machine Learning*, pp. 1278–1286.

Rumelhart, D.E., Hinton, G.E., Williams, R.J. *et al.* (1988). Learning representations by back-propagating errors. *Cognitive Modeling*, vol. 5, no. 3, p. 1.

Saxe, A.M., McClelland, J.L. and Ganguli, S. (2014). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *International Conference on Learning Representations*.

Schoenholz, S.S., Gilmer, J., Ganguli, S. and Sohl-Dickstein, J. (2017). Deep information propagation. *International Conference on Learning Representations*.

Stellenbosch University https://scholar.sun.ac.za

Sietsma, J. and Dow, R.J. (1991). Creating artificial neural networks that generalize. *Neural networks*, vol. 4, no. 1, pp. 67–79.

Silberer, C. and Lapata, M. (2012). Grounded models of semantic representation. In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 1423–1433. Association for Computational Linguistics.

Sompolinsky, H., Crisanti, A. and Sommers, H.-J. (1988). Chaos in random neural networks. *Physical Review Letters*, vol. 61, no. 3, p. 259.

Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958.

Sutskever, I., Martens, J. and Hinton, G.E. (2011). Generating text with recurrent neural networks. In: *International Conference on Machine Learning*, pp. 1017–1024.

Sutskever, I., Vinyals, O. and Le, Q.V. (2014). Sequence to sequence learning with neural networks. In: *Advances in Neural Information Processing Systems*, pp. 3104–3112.

Szegedy, C., Toshev, A. and Erhan, D. (2013). Deep neural networks for object detection. In: *Advances in Neural Information Processing Systems*, pp. 2553–2561.

Titsias, M. and Lázaro-Gredilla, M. (2014). Doubly stochastic variational Bayes for non-conjugate inference. In: *Internation Conference on Machine Learning*.

Vapnik, V. (1995). *The nature of statistical learning theory*. Springer science & business media.

Vincent, P., Larochelle, H., Bengio, Y. and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In: *International Conference on Machine Learning*, pp. 1096–1103.

Williams, C.K. and Rasmussen, C.E. (2006). *Gaussian processes for machine learning*. MIT Press Cambridge, MA.

Yang, G. (2019). Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*.