Understanding Gaussian Noise in Neural Networks



Alexander Dante Camuto

Supervisor: Chris Holmes

St Hugh's College

University of Oxford

Michaelmas 2021

A thesis submitted for the degree of Doctor of Philosophy

Abstract

Deep Learning (*read* neural networks) has emerged as one of the most exciting and powerful tools in the statistical machine learning toolbox. Neural networks over the past decade have defeated incumbent state of the art models in computer vision, natural language processing, and reinforcement learning; amongst a myriad of other fields. Though these achievements are impressive, neural networks remain surprisingly brittle, especially in the case of smaller network architectures. They can easily overfit, yielding poor performance on unseen data, and are remarkably susceptible to small perturbations and noise on their inputs. White noise on images has been shown to consistently alter the predictions of neural networks trained for classification. In a more targeted *adversarial setting*, where a hypothetical adversary has agency on the exact form of the input perturbation, these adverse effects are even more drastic.

As these networks are increasingly deployed on our devices and in critical decision making systems, it is in the community's broader interest to develop methods that not only allow smaller neural networks to generalise, but to develop methods that are robust to random and adversarially targeted perturbations on data.

An obvious candidate method to do this is to train neural networks with random perturbations on their training data, with the hopes that they are then able to make predictions that are robust to these perturbations. The limitations of this approach are well documented, as it only confers a strong degree of robustness to the exact perturbation distribution used at training time. As such, it doesn't confer a 'broad spectrum' robustness that one might hope. However, using these methods at *test-time* can confer this sought out certifiable robustness, one which we study at length in the context of *Variational Autoencoders*, a class of deep probabilistic model which experience Gaussian noise injections **by design**.

These methods also confer other overlooked benefits. For analytic simplicity, due to their finite moments, we focus our analysis on Gaussian Noise Injections (GNIs); but this analysis paves the way for the study of other forms of noise added to neural networks. As we demonstrate, in the case of GNIs during training, neural networks are regularised to learn a function that is biased towards lower frequencies in the Fourier domain. This tends to yield networks that not only generalise better to unseen data, but also have better calibrated predictions in the classification setting, meaning that the predictions they output are inherently more trustworthy. We also nuance this portrait, by showing that neural networks trained with *gradient descent* related methods also experience bias when trained with GNI, meaning that they will learn functions that aren't true minimizers of the objective function they are optimising for.

To present a unifying view with which to study GNI, this work ends with a framework using tools of Harmonic analysis, that explains both the lower frequency bias *and* the adversarial robustness conferred by GNI on neural networks.

This work presents a novel perspective from which to study deep probabilistic models (that of neural networks experiencing noise on their inputs), provides a comprehensive review of the positive and negative effects of training neural networks with GNI, and paves the way for the study of the effects of other types of noise on neural network training.

Prologue

Definition 0.1 (Noise)**.** *An unwanted signal or a disturbance.*

Example usage: Picture a lush quiet suburb, with chirping birds and sunshine. You are lying in your garden, in a sun-chair, the rustle of leaves above you is lulling you to sleep. As your eyelids are beginning to shut your neighbour suddenly revs up his cacophonous lawnmower. Your gentle afternoon slumber is shattered by the throaty rattle of the machine as it manicures the neighbouring lawn. The chirping birds have flown away; and the sun once gently kissing your skin is now giving you a headache. *Noise* has ruined your afternoon.

Now you're a Carnegie Hall audio-engineer. You and your team have meticulously spent your days and nights rigging the hall with recording equipment for the *Bavarian Radio Symphony Orchestra's* upcoming performance. The night goes well and at first glance you've produced an immaculate recording of their sublime rendering of Bach's Toccata and Fugue in D minor. A few days later you listen to it once again, and something doesn't feel right. You turn up the volume and you hear a subtle hiss and crackle in the right channel. Was it faulty equipment producing static noise ? A bad microphone connection ? An exposed wire ? These questions will haunt you as you fight tooth and nail to expunge noise from the recording. Noise has, once again, ruined your day.

Even psychologists and economists have a bone to pick with noise. Judges and doctors have been shown to give markedly different verdicts and diagnoses, *depending on the time of day* (Tversky and Kahneman, 1974). This unreliability, this noisiness in human decision making is something we might want to (reasonably) strangle at the source.

Why then would I study scenarios where we *willfully* add noise to neural networks involved in making critical predictions ? If noise is the great destroyer of reliable decision making, who in their right mind would do such a thing ? You might (understandably) call for me to be locked up in a padded cell for dedicating years of my life to the topic !

Well dear Reader, in the subsequent pages I hope to convince you that this was a worthwhile endeavour, and that intuitions about noise we carry from the broader world are not adequate for describing the effects of noise on neural networks. Indeed, rather counter-intuitively, noise provides an elegant and simple method that can help train neural networks that output more robust, accurate, and better calibrated predictions. It is perhaps, at least in machine learning, not the enemy we make it out to be.

Acknowledgments

I would like to thank my supervisor Chris Holmes, for taking a chance on me and for always encouraging my ideas. Thank you to Stephen Roberts, for not only mentoring me, but helping me through a rough landing when starting my degree.

Thank you also to Tom Rainforth and Brooks Paige for taking the time to mentor me.

Thank you to all my collaborators, whether we published together or not. Special thanks goes to Umut Şimşekli for introducing me to entirely new ways of thinking about Deep Learning.

Thank you to my friends Amna Askari, Eric Hambro, Thomas Chardin, and John Ferguson for endless discussions on Science and Technology. Thank you to Nikita Patel, Aaron Bhatoya, Ricardo de Luca e Tuma, Maria Lucey, Clara Dijkstra, Madeline White, Felix Keser, Leslie Labruto, Douglas Imrie for their friendship and support in Oxford and London.

Thank you to my girlfriend Amber Healey for her endless patience and support around conference deadlines.

Finally, this thesis would not have been possible without the widsom, support, kindness and patience of my collaborator and friend Matthew Willetts. This work would be immeasurably poorer had we not met.

CONTENTS

List of Figures ix

List of Tables xii

1 Introduction 1
- 1.1 Contributions . . . 6

2 Background 7
- 2.1 Neural Networks . . . 8
  - 2.1.1 Multi-layer perceptrons and Dense Layers . . . 9
  - 2.1.2 Convolutional networks . . . 10
  - 2.1.3 Back-propagation . . . 12
- 2.2 Stochastic Gradient Descent and Differential Equations . . . 13
- 2.3 Noise Injections in Neural Networks . . . 16
  - 2.3.1 Deep Latent Variable Models . . . 20

3 Explicit Regularisation in Gaussian Noise Injections 26
- 3.1 Introduction . . . 27
- 3.2 Background . . . 28
  - 3.2.1 Gaussian Noise Injections . . . 28
  - 3.2.2 Sobolev Spaces . . . 29
- 3.3 The Explicit Effect of Gaussian Noise Injections . . . 31
- 3.4 Fourier Domain Regularisation . . . 35
- 3.5 Related Work . . . 41
- 3.6 Conclusion . . . 41

4 Asymmetric Heavy Tails and Implicit Bias in Gaussian Noise Injections 43
- 4.1 Introduction . . . 44

4.2 Background . . . 46
4.2.1 Stable Distributions. . . . 46
4.2.2 Gaussian Noise Injections. . . . 49
4.3 The Implicit Effect of GNIs . . . 49
4.3.1 The Properties of the Accumulated Noise. . . . 50
4.3.2 Kurtosis of The Gradient Noise. . . . 52
4.3.3 Skewness of The Gradient Noise. . . . 54
4.4 An SDE Model for SGD with GNIs. . . . 54
4.4.1 Theoretical analysis of implicit bias. . . . 59
4.5 Further Experiments . . . 64
4.6 Conclusion . . . 65

5 Towards a Theoretical Understanding of the Robustness of Variational Autoencoders 67
5.1 Introduction . . . 68
5.2 Background . . . 70
5.2.1 Variational Autoencoders . . . 70
5.2.2 Adversarial Attacks for VAEs . . . 71
5.2.3 Disentangled VAEs . . . 71
5.3 Robustness of VAEs . . . 72
5.4 A Robustness Margin for VAEs . . . 73
5.4.1 r-robustness for VAEs . . . 74
5.4.2 Robustness to distortions in data-space . . . 75
5.5 Characterizing the Margin . . . 76
5.6 Empirical Investigations . . . 79
5.7 Robustness of Disentangled VAEs . . . 81
5.8 Conclusion . . . 83

6 Variational Autoencoders: A Harmonic Perspective 85
6.1 Introduction . . . 86
6.2 Background . . . 89
6.3 Gaussian Spaces and VAEs . . . 91
6.3.1 Gaussian Spaces and VAE decoders . . . 94
6.3.2 Gaussian Spaces and VAE encoders . . . 96
6.3.3 The KL and frequency content . . . 99
6.3.4 VAE Lipschitzness . . . 100
6.3.5 Potential Limitations . . . 101
6.4 Conclusion . . . 101

7 Conclusion 102
7.1 Extensions and Applications . . . 104

7.2 Implications . . . 105

A Appendix: The Explicit Regularisation in Gaussian Noise Injections 107
A.1 Technical Proofs . . . 108
A.1.1 Proof of Proposition 3.1 . . . 108
A.1.2 Proof of Theorem 3.2 . . . 108
A.1.3 Proof of Theorem 3.3 . . . 112
A.1.4 Regularisation in Regression Models and Autoencoders . . . 113
A.1.5 Regularisation in Classifiers . . . 113
A.2 Tikhonov Regularisation . . . 116
A.3 Measuring Calibration . . . 116
A.4 Classification Margins . . . 117
A.5 Model Capacity . . . 119
A.6 Additional Results . . . 120
A.7 Network Hyperparameters . . . 124

B Appendix: Asymmetric Heavy Tails and Implicit Bias in Gaussian Noise Injections 125
B.1 Additional Experimental Results . . . 126
B.2 Cost Functions . . . 129
B.2.1 Mean Square Error . . . 129
B.2.2 Cross Entropy Loss . . . 129
B.3 Other Potential Sources of the Skewness in the Gradient Noise . . . 129
B.4 Overview of the Assumptions . . . 130
B.5 Fractional Differentiation and the Approximation Scheme . . . 130
B.6 Metastability Analysis . . . 134
B.7 Technical Proofs . . . 139
B.7.1 Proof of Lemma 4.1 . . . 139
B.7.2 Proof of Theorem 4.1 . . . 142
B.7.3 Proof of Theorem 4.5 . . . 147
B.7.4 Proof of Theorem B.5.1 . . . 152
B.7.5 Proof of Corollary B.5.3 . . . 155
B.7.6 Proof of Theorem 4.6 . . . 156

C Appendix: Towards a Theoretical Understanding of the Robustness of Variational Autoencoders 160
C.1 Choosing r for r-robustness . . . 160
C.2 Proof of Theorem 5.1 . . . 163
C.3 Proof of Theorem 5.2 . . . 166
C.4 Empirical Calculation of the Bounds . . . 168
C.4.1 Estimating the minimum r . . . 168

C.4.2 Results . . . 168
C.4.3 Algorithm . . . 168
C.5 Estimating the Margin of Robustness . . . 169
C.6 -VAE Sensitivity Experiments . . . 170
C.7 Network Hyperparameters . . . 173

D Appendix: Variational Autoencoders: a Harmonic Perspective 174
D.1 Technical Proofs . . . 175
D.1.1 Proof of Proposition 6.1 . . . 176
D.1.2 Proof of Proposition 6.2 . . . 176
D.1.3 Proof of Theorem 6.3 . . . 177
D.1.4 Proof of Proposition 6.4 . . . 179
D.2 Multivariate Hermite Coefficients for a Gaussian measure with full Covariance . . . 180
D.3 Implementation Details . . . 181
D.4 Additional Results . . . 181
D.4.1 Fourier Transform of Gaussian Measure . . . 182
D.4.2 Fourier Spectrum of Multivariate VAEs . . . 182
D.4.3 Gaussian Spaces and VAE Lipschitzness . . . 183

E Bibliography 186

LIST OF FIGURES

1.1 Noise example. . . . 2

2.1 Fully connected architecture. . . . 10
2.2 Convolutional kernel movement across an input of dimensions width × height × depth and the resulting feature map generated. . . . 12
2.3 A fully connected network experiencing noise injections on its activations. . . . 17
2.4 DAG of a latent variable model with deterministic parameters. . . . 21
2.5 DAG of VAE generative model and variational posterior. . . . 23

3.1 Effect of GNIs injected throughout a network's activations. . . . 28
3.2 $R(\cdot)$ vs $\mathbb{E}_{\boldsymbol{\epsilon}}$ . . . 34
3.3 Networks trained with the explicit regularizer. . . . 35
3.4 The effect of GNIs on a network's Fourier spectrum. . . . 37
3.5 GNIs severely dampen the frequency content of neural network layers that are closest to the output. . . . 40

4.1 Illustration of the effect of GNIs *added* to a network's activations. . . . 45
4.2 Skewness and kurtosis *at initialisation* of the gradients noise accrued on networks weights during the backward pass for additive-GNIs. . . . 55
4.3 [left col.] sub-Weibull pdfs (r=0.8, 1.0, 2.0), and pdfs of MLE fitted $\mathcal{S}_\alpha$ and $\mathcal{N}$. [right col.] pdf in the tails ($x > 4$ and $x > 10$, note log y-axis). . . . 57
4.4 Replacing the implicit effect gradients with samples from α-stable distributions. . . . 58
4.5 Varying the levels of the implicit effect. . . . 58
4.6 The stationary distributions of (4.16) with $\alpha = 1.9$ (left) and $\alpha = 1.1$ (right). 60
4.7 Test-set loss of neural networks trained with the explicit regulariser approximation. . . . 63
4.8 Illustration of how the kurtosis and skewness of the implicit effect degrades test-set performance. . . . 64

5.1 Reconstructions under attack for robust and non-robust VAEs. 69
5.2 Illustration of r-robustness in a VAE. 74
5.3 Illustration of the margin $R^r_{\mathcal{X}}(\mathbf{x})$, which is defined in the **input** space $\mathcal{X}$. . . 75
5.4 *Maximum damage* adversarial attacks on multiple VAEs trained on MNIST (a), fashion-MNIST (b), and CIFAR10 (c). 76
5.5 $R^r_{\mathcal{X}}(\mathbf{x})$ for four VAEs of varying robustness trained on MNIST. 77
5.6 Empirically estimated $R^r_{\mathcal{X}}(\mathbf{x})$ against the bound for $R^r_{\mathcal{X}}(\mathbf{x})$ defined in Theorem 1, ignoring higher order terms. 78
5.7 Ablation study on the bounds defined by Theorem 5.1. 79
5.8 For β-VAEs trained with $\beta \in \{0.1, 1, 10\}$ we show in consecutive rows first the original data point, a perturbed version made by maximum damage adversarial attacks, and then the reconstruction given by the model. 81
5.9 (a) distribution of the numerically estimated $R^r_{\mathcal{X}}(\mathbf{x})$ across the MNIST dataset. (b) likelihood of the original input given a maximum damage adversarial attack as in Eq (5.5). (c) and (d) show that the encoder variance increases and the encoder Jacobian norm ($||\mathbf{J}^{\mu}_{\phi}(\mathbf{x})||_F$) decreases as β increases. 82

6.1 VAE sinc function reconstructions under varying encoder variance. 87
6.2 FFT of the VAE reconstructions in Figure 6.1. 96
6.3 2D-FFT of VAE reconstructions under varying encoder variance. 97
6.4 FFT of μ_ϕ of VAEs trained on $\text{sinc}(5t) + \mathcal{N}(0, \sigma^2\mathbf{I})$. 99
6.5 Positive components of Figure 6.4 with variable β penalisation on the KL term of the ELBO. 100

A.1.1 Samples of heatmaps of 10 by 10 matrices $\mathbf{J}_k^\intercal\mathbf{J}_k$ (k indexing over layers) for 2-layer MLPs and convolutional networks. 115
A.4.2 Here we show distribution plots of $\mathbf{J}_0$ for 2-layer MLPs trained on CIFAR10 (a) and SVHN (b) for models trained with no noise (Baseline), models trained with noise on their inputs (GNI Input), models trained with noise on all their layers (GNI All Layers). 118
A.6.3 Test set loss for the regression dataset Boston House Prices (BHP) for MLPs trained with $R(\cdot)$ and GNIs for $\sigma^2 = 0.1$. We compare to a non-noised baseline (Baseline). 120
A.6.4 We use small variance noise injections and show that the $R(\cdot)$ (Exp Reg) in equation (3.11) and (3.14), induces the same trajectory through the loss landscape as GNIs (Noise). 120
A.6.5 Illustration of the loss induced by the $R(\cdot)$ for classification detailed in equation (3.14) for convolutional and MLP architectures, and for ReLU and ELU non-linearities. 121
A.6.6 Measure of a model's sensitivity to noise by adding noise of variance α^2 to data and measuring the resulting model accuracy given this corrupted test data. 122

A.6.7Illustration of how Gaussian noise (Noise) *additions* improve calibration relative to models trained without noise injections (Baselines) and how $R(\cdot)$ (Exp Reg) also captures some of this improvement in calibration. 123

B.1.1Skewness and kurtosis *at initialisation* of the noise accumulated on network activations at each layer i 127
B.1.2Here we show the same plots as in Figure B.1.1 but for multiplicative-GNIs. 127
B.1.3Here we show the same plots as in Figure 4.2 but for multiplicative-GNIs. 127
B.1.4We model $\nabla E_{\mathcal{L}}(\cdot)$ as being drawn from some α-stable distribution $\mathcal{S}_\alpha$ and estimate the tail-index α and skewness θ. 128
B.3.5Here we reproduce Figure 4.2 but with no non-linearities. 129

C.2.1Illustration of the boundary R we are measuring in $\mathcal{Z}$. 166
C.4.2The minimum r for which $p(||\Delta(\mathbf{x})||_2 \leq r) = 0.5$ increases with β and τ, where β is the penalty applied to the KL in β-VAEs and τ is an offset added to the encoder standard deviation $\boldsymbol{\sigma}_\phi(\mathbf{x})$. 168
C.6.3illustration that β-VAEs, trained on MNIST, with higher β penalties generalise better and are less sensitive to input perturbations. 170
C.6.4Illustration that β-VAEs, trained on fashion-MNIST, with higher β penalties generalise better and are less sensitive to input perturbations. 171
C.6.5Illustration that β-VAEs, trained on CIFAR10, with higher β penalties generalise better and are less sensitive to input perturbations. 172
C.6.6Reconstructions of noisy data for VAEs trained with $\beta \in \{0.1, 1, 10\}$ on Fashion-MNIST. 172

D.4.1[left] Fourier transform of the Gaussian measure $\mathcal{N}(0, \sigma^2)$ for varying σ. [right] Fourier transform of the Gaussian measure $\mathcal{N}(\mu, 1)$ for varying μ. . 182
D.4.2We train VAEs on multivariate data consisting of 5 sinusoids, with a 1 dimension latent space $\mathcal{Z}$. **[top row]** We show plots of the regressed sinusoids for different values of σ_ϕ. **[bottom row]** FFT for each regressed sinusoid. 182
D.4.3Both plots show the relative log likelihood degradation resulting from a 'maximum-damage' (Camuto et al., 2020) adversarial attack against the maximum attack norm C. As σ, the variance of isotropic Gaussian noise added to data, and σ_ϕ, the fixed posterior variance, increase, the degradation lessens as C increases, highlighting that both these parameters improve robustness. Shading corresponds to the variance of the likelihood degradation over 25 points from CIFAR10. 184

LIST OF TABLES

D.4.1 The Lipschitz constants (L) of VAEs' **[left]** decoder networks (g_θ) when trained with fixed encoder scale (σ_ϕ) and **[right]** encoder networks (μ_ϕ) when trained with σ-scale Gaussian noise injection on inputs and fixed encoder variance σ_ϕ=0.5. We train fully-connected VAEs with the Sigmoid activation in their networks; on sinc($5t$) ($\dim(\mathcal{Z}) = 1$), on CelebA ($\dim(\mathcal{Z}) = 64$), and CIFAR10 ($\dim(\mathcal{Z}) = 64$). Each network has 3 dense layers each with 256 units. As σ_ϕ decreases the Lipschitz constant (L) increases, as predicted by Proposition 6.4. As σ decreases, the Lipschitz constant (L) of the encoder mean (μ_ϕ) increases. ($\pm$) is the std. dev. over 3 random seeds. 183

CHAPTER 1

INTRODUCTION

1.1 Contributions 6

2.1 Neural Networks 8

2.1.1 Multi-layer perceptrons and Dense Layers 9

2.1.2 Convolutional networks 10

2.1.3 Back-propagation 12

2.2 Stochastic Gradient Descent and Differential Equations 13

2.3 Noise Injections in Neural Networks 16

2.3.1 Deep Latent Variable Models 20



Figure 1.1: We have an image of a panda (left) and that same image corrupted with white noise (right). The image on the right can clearly still be identified as a panda. Nevertheless attacks of this form have been surprisingly effective against neural networks (Szegedy et al., 2014; Shamir et al., 2019; Goodfellow et al., 2015; Papernot et al., 2016; Moosavi-Dezfooli et al., 2016; Franceschi et al., 2018). Figure taken from (Cohen et al., 2019).

Neural networks have entered the mainstream. They power the recommendation systems that underpin e-commerce websites. They're behind the ever-present voice assistants in our homes. Soon they might even be behind the steering wheels of our cars. However, ask someone when they last interacted with a neural network, and most people will stare back at you blankly. They operate out of view in the shadows of our phones, computers, and factories; working subtly to improve the efficiency of many systems.

Though they are ubiquitous, neural networks remain remarkably fragile. Some of the most sophisticated deep learning models can be broken by simply adding small pertubations to their inputs (Szegedy et al., 2014; Shamir et al., 2019; Goodfellow et al., 2015; Papernot et al., 2016; Moosavi-Dezfooli et al., 2016). Perturbations that would not fool a human break neural network predictions (see Figure 1.1).

Not only are the networks fragile, they're also prone to overfit in supervised settings, meaning that any performance increases on training data actually induce *worse* performance on held-out test data. As an extreme example, consider the case where the number of neural network parameters is equal to the number of training samples. The neural network can effectively 'memorise' the training set, fitting to the noise and residual variation in the training data, and thus being unable make accurate predictions for data the network

has not been trained with (Bishop, 2006). An interesting, rather counter-intuitive, phenomena occurs for neural networks that have more parameters than data-points. Beyond this threshold the generalisation gap (the difference between test set loss and training set loss) begins to diminish once more as the number of parameters increases, eventually falling below the minimum gap obtained in the small number of parameter regime (anything less than the number of datapoints) (Nakkiran et al., 2019).

For applications where such large networks are unsuitable; such as applications on low-power devices, low-memory devices, or that require fast-predictions; improving test-set performance without increasing the number of parameters is of critical importance. In the traditional statistics setting, methods that reduce the gap between training and test-set performance are called *regularisation methods*.

If one wanted to address the first flaw, the inherent brittleness of networks to perturbations, one could train a network under noised and perturbed data expecting that it will learn to deliver reliable predictions even under noisy conditions. The caveat to this idea is that this will only work if the noise at test-time has the same distribution as the noise used during training. The aim of this work is to show that the benefits and effects of injecting noise into neural networks at train and test-time go much deeper than providing very limited protection to perturbations. Indeed they provide a simple mechanism by which practitioners can improve the generalisation and the robustness to adversarial perturbations of neural networks. As we demonstrate they constitute a simple regularisation method that can improve test-set performance and the calibration of networks when used at training time. But they can also improve the adversarial robustness of networks *when used at test-time*.

More formally, noise injections are a family of methods that involve adding or multiplying samples from a noise distribution, typically an isotropic Gaussian or a Bernoulli distribution, to the weights and activations of a neural network during training. The benefits of such methods, in terms of generalisation to unseen data, are well documented (Srivastava et al., 2014; Kingma et al., 2015; Poole et al., 2014). Though the regularisation conferred by such noise injections can be observed empirically, and there have been many studies on the benefits of noising data (Bishop, 1995; Cohen et al., 2019; Webb, 1994),

there have been few studies on understanding the benefits of methods that inject noise *throughout* a network.

Because of their simple properties (such as finite higher moments) and of their widespread appearance in real-world systems [1] we focus our attention and analysis on noise injection methods with Gaussian noise.

By their nature these noise injections are ripe for studying under two lenses: the *explicit* and *implicit* effects, introduced by Wei et al. (2020) for Bernoulli dropout. The *explicit* effect is the added term we obtain to the loss function when we marginalise out the noise we have injected. We present this explicit effect for Gaussian noise in our first presented work (§3).

The *implicit* effect is then the effect of the remainder from the marginalisation performed to extract the explicit effect. Though Wei et al. (2020) present empirical evidence that this effect is significant – they do not build a theoretical framework to explain how and why this remainder may have an effect. In our third work (§4) we construct such a framework by focusing on the way noise injections alter the dynamics of Stochastic Gradient Descent (SGD) and how these altered dynamics then impact neural network performance.

We also show that the stochastic layers of deep probabilistic models; such as Variational Autoencoders (VAEs) are directly related to Gaussian noise injections. This observation forms the crux of our second and fourth presented works (§5 & §6).

To summarise, this thesis is structured as follows:

- We begin with a literature review that presents context to understand the heart of the presented material (§2).
- Subsequent to this we study the explicit effect of Gaussian noise injections, which is the effect they have on neural networks when we marginalise out the injected noise (§3).

[1] In conversations with companies manufacturing analog processors to accelerate deep learning, we've discovered that analog processors inadvertently inject Gaussian noise into neural network parameters and activations. As a result, understanding the effects this noise has on neural network training is of vital importance for such manufacturers.

- To nuance the benefits of Gaussian noise we also show that these injections can induce *bias* that degrades the performance of models trained with stochastic gradient descent (§4).

- To increase the applicability of our analysis we show that deep probabilistic models with Gaussian priors, such as Variational Autoencoders (VAEs), inadvertently experience Gaussian noise on the inputs of their constituent networks, and that this noise improves the adversarial robustness of these models (§5).

- Finally we end with a more holistic perspective that unifies and simplifies much of the previous analysis. We develop a set of tools from the functional analysis of Gaussian Spaces, a type of measure space, to study VAEs and more broadly to study neural networks trained with Gaussian noise on their inputs (§6).

The four works presented in the subsequent have all been published. The first (§3) was published at The Annual Conference on Neural Information Processing Systems (Neurips), the second (§4) at The International Conference on Machine Learning (ICML), and the third and fourth work (§5 & §6) at The Society for AI and Statistics (AISTATS). Note that this thesis is presented in **integrated format**. The works are presented reformatted using the current thesis formatting but are word-for-word reproductions of the published works and as such each chapter includes its own abstract and detailed background sections.

Other works written during my DPhil candidacy, influenced by and building upon the subsequent presented works, include:

- *Improving VAEs' Robustness to Adversarial Attack*, M Willetts, **A Camuto**, T Rainforth, S Roberts, C Holmes, International Conference on Learning Representations (ICLR), 2021

- *Certifiably Robust Variational Autoencoders*, B Barrett, **A Camuto**, M Willetts, T Rainforth, AISTATS, 2022

- *Learning Bijective Feature Maps for Linear ICA*, **A Camuto**, M Willetts, B Paige, C Holmes, S Roberts, AISTATS, 2021

- *Fractal Structure and Generalization Properties of Stochastic Optimization Algorithms*, **A Camuto**, G Deligiannidis, MA Erdogdu, M Gürbüzbalaban, U Şimşekli, Lingjiong Zhu, The Annual Conference on Neural Information Processing Systems (Neurips), 2022

1.1 Contributions

Though the ideas for the subsequent works were mine, they would not have been possible without the help and support of many collaborators.

In §3 the theory, experiments, and writing were my work. Matthew Willetts, Umut Şimşekli, Stephen Roberts, and Chris Holmes contributed guidance on writing and theory.

In §5 the theory, experiments, and figures were mostly my doing. Theorem 5.2 and some of the writing were the work of Matthew Willetts, Tom Rainforth, Stephen Roberts, and Chris Holmes.

In §4 Theorem 4.6, Theorem 4.5 (and the very expansive associated Appendices), and some of the writing were the work of Xiaoyu Wang, Lingjiong Zhu, Chris Holmes, Mert Gürbüzbalaban, and Umut Şimşekli. All other theorems, experiments, figures, and most of the writing were my work.

In §6 the theory, experiments, and writing were my work. Matthew Willetts contributed guidance on writing and theory.

CHAPTER 2

BACKGROUND

To provide a fuller context for understanding the four works that constitute the body of this thesis, in the subsequent we present a review of:

- Neural networks and their different architectures.
- Subsequently we present how neural networks are trained using stochastic gradient descent (SGD) and the connections between SGD and differential equations.
- We then show how noise injections are performed on neural networks and how they have been studied prior to this, particularly in the adversarial attack setting.
- Finally we end with a review of deep probabilistic models, focusing on Variational Autoencoders (VAEs), and how their constituent networks experience Gaussian noise injections.

Given this thesis is in integrated format, the main body of the thesis also includes separate background and exposition sections for each work. These provide additional mathematical backgrounds and literature reviews important for understanding each individual chapter.

2.1 Neural Networks

Neural networks are sets of connected computational 'units', which, though they are not powerful in isolation, can accomplish incredibly sophisticated tasks *in aggregate*. This modularity and composability is one of their key strengths, in that defining a more complex model simply requires adding more units of computation.

Other models, such as tree-based models like Random Forests also have this characteristic. However, what these models lack is a library of incredibly rich architectures, which are specific patterns and arrangements of the computational units, to draw from. As we will briefly overview, these specialized architectures have allowed neural networks to excel in many domains.

To understand neural networks, one must first understand their constituent computational units. Each of these units is at its core, a generalised linear model (GLM), or 'perceptron' in 1960s terminology (Rosenblatt, 1963). In the generalised linear setting, which is a *supervised* learning setting, we have a dataset where each datapoint (or 'covariate') $\mathbf{x} \in \mathbb{R}^n$ has an associated label $\mathbf{y} \in \mathbb{R}^m$. Our model has weights $\mathbf{w} \in \mathbb{R}^{m \times n}$, a bias $\mathbf{b} \in \mathbb{R}^m$, an *element-wise* activation function κ (i.e in our notation $\kappa(\mathbf{x})$ is equivalent to $[\kappa(x_1) \dots \kappa(x_n)]^T$), and our objective is to minimize the log-likelihood of the labels $\mathbf{y}$, given data $\mathbf{x}$, with respect to the parameters $\{\mathbf{w}, \mathbf{b}\}$ to find the optimal parameters $\{\mathbf{w}^*, \mathbf{b}^*\}$ (Haykin, 1999; Bishop, 2006). With a training dataset composed of N data-label pairs $\mathcal{D}$, we minimize the expected log-likelihood over $\mathcal{D}$

$$\mathbf{w}^*, \mathbf{b}^* = \min_{\mathbf{w},\mathbf{b}} \mathbb{E}_{p(\mathcal{D})} \left[\log p(\mathbf{y}|\mathbf{x}; \mathbf{w}, \mathbf{b})\right] \tag{2.1}$$

$$\text{where} \qquad p(\mathbf{y}|\mathbf{x}; \mathbf{w}, \mathbf{b}) = p(\mathbf{y}|\kappa(\mathbf{w}\mathbf{x} + \mathbf{b})), \tag{2.2}$$

and $p(\mathcal{D})$ is the empirical distribution of the training dataset $\mathcal{D}$, i.e of the data-label pairs $(\mathbf{x}, \mathbf{y})$. The manner in which this optimisation is performed is covered in §2.2. If κ is the sigmoid function $\left(\kappa(x) = \frac{1}{1+e^{-x}}\right)$ and the conditional probability $p(\mathbf{y}|\mathbf{x}; \mathbf{w}, \mathbf{b})$ is the Bernoulli distribution, we recover the case of binary classification, used when $y \in \{0, 1\}$. If κ is the identity function, and the conditional probability $p(\mathbf{y}|\mathbf{x}; \mathbf{w}, \mathbf{b})$ is some member of the exponential family, we recover the case of linear regression where y can lie in $\mathbb{R}$.

Note that for notational simplicity the bias can be represented by augmenting the covariates with a '1', such that $\mathbf{x}$ is now the concatenation of the original covariates and a '1' element. The weights now lie in $\mathbf{w} \in \mathbb{R}^{m\times(n+1)}$ and we have that,

$$\mathbf{w}^* = \min_{\mathbf{w}} \mathbb{E}_{p(\mathcal{D})} \left[\log p(\mathbf{y}|\mathbf{x};\mathbf{w})\right] \tag{2.3}$$

$$\text{where} \qquad p(\mathbf{y}|\mathbf{x};\mathbf{w}) = p(\mathbf{y}|\kappa(\mathbf{w}\mathbf{x})). \tag{2.4}$$

To lessen burden on notation, this is a perspective we take in the main chapters of this thesis and in the subsequent exposition on multi-layer perceptrons.

2.1.1 Multi-layer perceptrons and Dense Layers

GLMs are not particularly powerful models, however they are inherently composable, in that the outputs of one GLM can constitute the covariates $\mathbf{x}$ of another (Rumelhart et al., 1986; Haykin, 1999; Bishop, 2006). By stacking multiple GLMs in parallel as 'layers', and by feeding the outputs of one layer into another GLM layer, we obtain a dense 'network' of connections and learnable parameters, which in aggregate form a *neural network*. More formally, if we denote the output of one set of GLM, formally the i^{th} layer 'activations', as $\mathbf{h}_i \in \mathbb{R}^{n_i}$ (where each set of activations is concatenated with a 1 element to represent bias terms), we have the following recursive relationship:

$$\mathbf{h}_i(\mathbf{x}) = \kappa\left(\mathbf{W}_i \mathbf{h}_{i-1}(\mathbf{x})\right), \tag{2.5}$$

$$\text{where } \mathbf{h}_0 = \mathbf{x}, \ \mathbf{W}_i \in \mathbb{R}^{n_i \times n_{i-1}},$$

and the model's D weights are now divided into L layers: $\mathbf{w} \equiv \{\mathbf{W}_1, ..., \mathbf{W}_L\} \in \mathbb{R}^D$, and $\mathbf{W}_i, i = 1 \dots L$ are matrices. We now learn the parameters (split into L layers) by minimizing

$$\mathbf{w}^* = \min_{\mathbf{w}} \mathbb{E}_{p(\mathcal{D})} \left[\log p(\mathbf{y}|\mathbf{x};\mathbf{w})\right] \tag{2.6}$$

$$\text{where} \qquad p(\mathbf{y}|\mathbf{x};\mathbf{w}) = p(\mathbf{y}|\kappa\left(\mathbf{W}_L \mathbf{h}_L(\mathbf{x})\right)), \ \mathbf{W}_L \in \mathbb{R}^{m\times n_L},$$

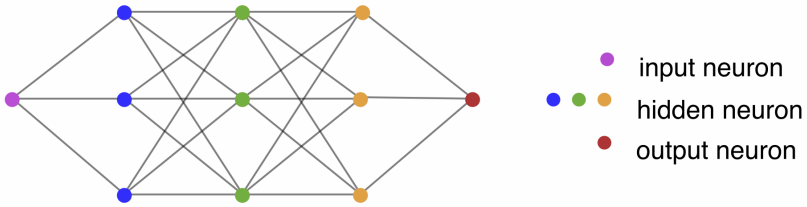


Figure 2.1: Fully connected architecture. Each layer of neurons is densely connected to the next by weights (represented as lines).

and $\mathbf{h}_L(\mathbf{x})$ effectively contains the output of the computations executed by all previous layers in the neural network and m is the dimensionality of the output and labels $\mathbf{y}$.

In the setup we've described, the tunable weights at each layer are multiplied to each of the previous layer's activations. For a visual illustration of this see Figure 2.1. This sort of architecture is suitable for analysing covariates that are independent, or that are not sequential in time (such as time-series data), or spatially organised (such as image data) (Bishop, 2006). The modularity and flexibility of neural networks means that other architectures that are better suited to these problems have been developed. The bulk of this work, for simplicity of theoretical analysis, focuses on dense layers. Nevertheless for completeness we present a brief overview of other architectures we use, in particular convolutional networks.

2.1.2 Convolutional networks

Images are one of the most prevalent forms of digital data. The widespread distribution of smartphones with cameras has created an explosion in the number of images we generate and in turn has created opportunities for applications that can process images and generate medical classifications, silly-filters, or real time image stabilisation (among many many others). At the core of this image processing revolution lie convolutional neural networks (ConvNets), ingenious twists on the MLP architecture above which are better suited to spatially structured data like images (Fukushima, 1980; LeCun et al., 1989, 1998b; Krizhevsky et al., 2012).

Images are typically 3-dimensional data structures consisting of a width, height, and depth – or 'number of channels'. In the case of the commonly used RGB colour encoding, each unique 'pixel' in the set of width $\times$ height pixels of the images is represented by 3 integers (one for each colour channel) in the range $[0, 255]$.

In a convolutional layer, Each neuron consists of a weight kernel and a non-linear activation function (as before). This kernel is a set of weights, multiplied as a dot product with input data, the output of which is then fed through a non-linear activation function. This process is very similar to that of an MLP that we described before. The core difference between MLPs and ConvNets lies in the fact that:

- Each weight kernel of dimensions $w \times h \times d$ (d is often the same as the depth of the image) is only multiplied (as a dot product) to a subset of the image of dimensions width $\times$ height $\times$ depth. Convolutional parameters are thus more *sparsely connected* to inputs (compared to the densely connected weights of MLPs).

- This kernel is then shifted around the image, whereby each movement consists of a shift across s pixels in the width, depth or height of the image, every time performing a dot product operation between the weight kernel and a portion of the image of dimensions $w \times h \times d$. s is called the 'stride' of the kernel.

- Each neuron thus produces a feature map of width $(\text{width} - w)/s + 1$ and height $(\text{height} - h)/s + 1$ (see Figure 2.2 for an illustration). As the depth of the kernel is in practice often the same as that of the processed image, the number of channels of the resulting output is the same.

In practice, note that the input covariates can be padded (meaning we add data around the original covariates, such as $0s$ or $1s$), to accommodate kernel-size and stride lengths that wouldn't fit neatly within the original covariate dimensions. A convolutional layer is then a set of k kernels, *each* producing a feature map, such that output of a convolutional layer is a set of k maps of varying dimensions. Features maps can then serve as the inputs to other convolutional layers or as inputs to 'pooling' layers, which reduce the dimensionality of feature maps by extracting summary statistics. Pooling layers operate in the same manner as a convolutional layer, but instead of having a kernel with learnable

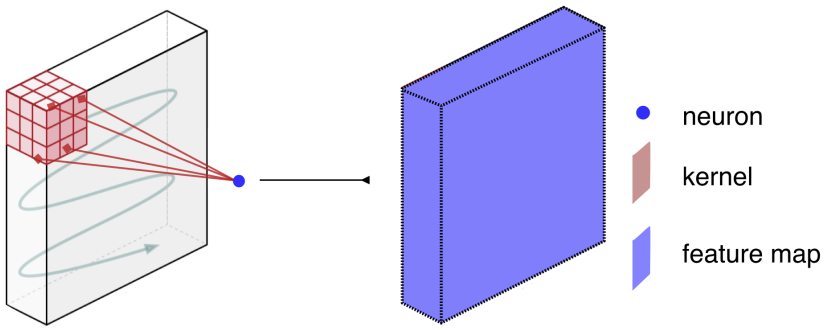


Figure 2.2: Convolutional kernel movement across an input of dimensions width × height × depth and the resulting feature map generated.

parameters, they extract a single statistic, such as the mean, or the max over the patch of size $w \times h \times d$ (LeCun et al., 1989, 1998b).

2.1.3 Back-propagation

A core property of neural networks is that their layers can not only be composed together to form a complex structure of computations that feed into one another; their structure is such that, given a sufficiently regular loss function with respect to the network output, the derivative of said loss with respect to network parameters can be computed fairly simply. More formally, if we define a neural network as a mapping $f_{\mathbf{w}} : \mathbb{R}^n \to \mathbb{R}$, parameterised by weights $\mathbf{w}$, for a function $\mathcal{L}$, that is once differentiable with respect to its inputs, then we can calculate:

$$\nabla_{\mathbf{w}} \left[\mathcal{L}(f_{\mathbf{w}}(\mathbf{x})) \right], \text{where } \mathbf{x} \text{ is a covariate.}$$

The calculation of these derivatives is rather simple, in that we can recursively apply the derivative chain rule to each layer, starting with the output (L^{th}) layer (Rumelhart et al., 1986; Bishop, 2006):

$$\frac{\partial \mathcal{L}(f_{\mathbf{w}}(\mathbf{x}))}{\partial W_{L,j,k}} = \frac{\partial \mathcal{L}(f_{\mathbf{w}}(\mathbf{x}))}{\partial h_{L,k}} \frac{\partial h_{L,k}}{\partial W_{L,j,k}},$$

where k indexes over the activations of the output layer and j indexes over the activations of the $(L-1)^{\text{th}}$ layer. Because the metric $\mathcal{L}$ is once differentiable with respect to (w.r.t) its inputs, the first term is easy to calculate. Similarly, if we constrain our activation functions to be once differentiable w.r.t its inputs, the second term can be calculated

with ease. From this we obtain a recursive relationship for the calculation of parameter derivatives:

$$\frac{\partial \mathcal{L}(f_{\mathbf{w}}(\mathbf{x}))}{\partial W_{i,j,k}} = \frac{\partial \mathcal{L}(f_{\mathbf{w}}(\mathbf{x}))}{\partial h_{i,j}} \frac{\partial h_{i,k}}{\partial W_{i,j,k}} = \sum_{l=1}^{N_{i+1}} \underbrace{\frac{\partial \mathcal{L}(f_{\mathbf{w}}(\mathbf{x}))}{\partial h_{i+1,l}}}_{\text{already calculated for } (i+1)^{\text{th}}\text{layer}} \underbrace{\frac{\partial h_{i+1,l}}{\partial h_{i,k}} \frac{\partial h_{i,k}}{\partial W_{i,j,k}}}_{\text{easy to calculate}}.$$

We can efficiently calculate the derivative of each layer's parameters, starting with the final output layer, as the computations used for the next layer directly determine the gradients for the current layer.

This ability to easily calculate the parameter derivatives forms the core of how modern neural networks are trained, as the networks can then learn and iteratively update their parameters using optimisation algorithms that are related to *Gradient Descent*. We use the recursive nature of gradient descent when ascertaining the kurtosis and skewness of the gradient noise induced by GNIs in §4.

2.2 Stochastic Gradient Descent and Differential Equations

Training a neural network for classification or regression tasks is carried out by tuning the parameters of the network to optimise the marginal likelihood of a set of labels $\mathbf{y}$ given features $\mathbf{x}$. With a training dataset composed of data-label pairs $\mathcal{D} \equiv \{(\mathbf{x}_0, \mathbf{y}_0)...(\mathbf{x}_N, \mathbf{y}_N)\}$ and a feed-forward neural network with D parameters divided into L layers: $\mathbf{w} \equiv \{\mathbf{W}_1, ..., \mathbf{W}_L\} \in \mathbb{R}^D$, our objective is to minimise the expectation of negative log likelihood (Bishop, 2006) over the empirical data distribution of data-label pairs $(\mathbf{x}, \mathbf{y})$, $p(\mathcal{D})$:

$$\mathcal{L}(\mathcal{D}; \mathbf{w}) = -\mathbb{E}_{p(\mathcal{D})} \log p_{\mathbf{w}}(\mathbf{y}|\mathbf{x}). \tag{2.7}$$

Under maximum likelihood estimation, our aim is to find the optimal set of parameters $\mathbf{w}^*$ satisfying:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathcal{L}(\mathcal{D}; \mathbf{w}). \tag{2.8}$$

Stochastic Gradient Descent (SGD) and it's variants are the most prevalent optimisation routines that underpin the training of very large state-of-the-art neural networks.

Under SGD, we estimate equation (4.1) by sampling a mini-batch of data-label pairs $\mathcal{B} \subset \mathcal{D}$,

$$\mathcal{L}(\mathcal{B}; \mathbf{w}) = -\mathbb{E}_{\mathbf{x},\mathbf{y}\sim\mathcal{B}} \log p_{\mathbf{w}}(\mathbf{y}|\mathbf{x}) \approx \mathcal{L}(\mathcal{D}; \mathbf{w}), \tag{2.9}$$

and use the gradients of this simpler object to perform optimisation. SGD optimises equation (4.1) and approximates $\mathbf{w}^*$ using iterative parameter updates. At training step k we can write this (discrete) update as:

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \rho \nabla_{\mathbf{w}^k} \mathcal{L}(\mathcal{B}; \mathbf{w}^k), \tag{2.10}$$

where ρ is the step-size for updates, also known as the network's learning rate (Robbins and Monro, 1951; Ruder, 2016).

When we train *without* sampling batches, and train the network with $\mathcal{D}$ at each time-step with updates $\mathbf{w}^{k+1} = \mathbf{w}^k - \rho \nabla_{\mathbf{w}^k} \mathcal{L}(\mathcal{D}; \mathbf{w}^k)$, we have the following continuous time approximation to gradient descent, by taking the step-size $\rho \to 0$,

$$d\mathbf{w}^t = -\nabla \mathcal{L}(\mathcal{D}; \mathbf{w}^t).$$

When we consider *stochastic* gradient descent, with a batching process $\mathcal{B}$, we must consider the noise that is added to the above differential equation by the sampling of batches. This noise stems from the inherent stochasticity in approximating the 'true' gradient over the dataset $\nabla_{\mathbf{w}^k} \mathcal{L}(\mathcal{D}; \mathbf{w}^k)$ with that of a mini-batch $\mathcal{B}$, $\nabla_{\mathbf{w}^k} \mathcal{L}(\mathcal{B}; \mathbf{w}^k)$. We denote this noise:

$$U_k(\mathbf{w}) = \nabla_{\mathbf{w}^k} \mathcal{L}(\mathcal{B}; \mathbf{w}^k) - \nabla_{\mathbf{w}^k} \mathcal{L}(\mathcal{D}; \mathbf{w}^k). \tag{2.11}$$

Because of this noise, the continuous time approximation becomes a stochastic differential equations (SDE) and there have been many studies modelling the dynamics of SGD using SDEs under different noise conditions (Agapiou et al., 2014; Şimşekli et al., 2019a,b; Nguyen et al., 2019; Şimşekli et al., 2020; Gao et al., 2018; Jastrzębski et al., 2017). By far, the most prevalent assumption is to model the gradient on the network parameters as having multivariate Gaussian (i.e Brownian motion) noise: $U_k(\mathbf{w}) \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$. This is rationalised by the fact that by the the central limit theorem, the sum of estimation errors

in equation (2.11) is approximately Gaussian. Using this we can rewrite the parameter update for SGD as:

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \rho\mathcal{L}(\mathcal{D}; \mathbf{w}^k) + \rho U_k(\mathbf{w}). \tag{2.12}$$

We obtain the following continuous stochastic differential equation to approximate the gradient updates (Welling and Teh, 2011; Şimşekli et al., 2019b; Mandt et al., 2016):

$$d\mathbf{w}^t = -\nabla\mathcal{L}(\mathcal{D}; \mathbf{w}^t) + \sqrt{\nu\sigma^2}d\mathbf{B}^t, \tag{2.13}$$

where $\mathbf{B}^t$ is the *standard* d-dimensional Brownian motion, σ is the assumed noise variance for U.

Recent research suggests that for *large* neural networks the aforementioned Gaussian assumption may not always be appropriate. Instead, in many settings, gradient noise is heavy-tailed (Şimşekli et al., 2019b; Şimşekli et al., 2020). We must now apply the generalised central limit theorem (Gnedenko and Kolmogorov, 1954; Johnson and Samworth, 2005) to characterise $U_k(\mathbf{w})$. This theorem, as the name suggests, is a generalization of the Central Limit Theorem, that states that for a sequence of independent and identically distributed (i.i.d.) random variables whose distribution has a power-law tail with index $0 < \alpha < 2$, the normalised sum converges to a heavy-tailed distribution called the α-stable distribution ($\mathcal{S}_\alpha$) as the number of summands grows. If $\alpha \geq 2$ we recover the central limit theorem, whereby the sum converges to an $\alpha\mathcal{S}$ distribution with stability parameter α equal to 2, which corresponds to a Gaussian distribution.

We offer a more complete introduction to $\alpha\mathcal{S}$ distributions in §4, but broadly note that as the stability parameter α decreases the distribution becomes increasingly heavy-tailed. Assuming that our gradient noise is symmetric we have $U_k(\mathbf{w}) \sim \mathcal{S}\alpha\mathcal{S}(\sigma^2, \alpha)$, where $\mathcal{S}\alpha\mathcal{S}(\sigma^2, \alpha)$ denotes the D-dimensional symmetric alpha stable distribution with variance σ^2 [1] across each dimension and direction dependent tail-parameter $\alpha = (\alpha_i, 1 \leq i \leq D)$, such that each coordinate can have its idiosyncratic parameter α_i. The stochasticity of SGD, before modelled by *Brownian* motion, is instead better modelled by *Lévy* motion

[1] This scale parameter can also be direction dependent, but for notational simplicity, and later analytic simplicity in §4 we use a scalar.

(Şimşekli et al., 2019b),

$$d\mathbf{w}^t = -\nabla\mathcal{L}(\mathcal{D};\mathbf{w}^t) + \rho^{\frac{\alpha-1}{\alpha}}\sigma d\mathbf{L}_\alpha^t, \tag{2.14}$$

where $\mathbf{L}_\alpha^t$ is a D-dimensional α-stable Lévy motion. Empirically, SGD with small batch sizes and large learning rates tends to favour 'wider' minima in the loss function energy landscape (Jastrzębski et al., 2017; Keskar et al., 2019). In the context of Lévy motion, this tendency to land in wider minima can be described using metastability analysis, under which we measure the minimum time, the *first exit time* τ, for the network parameters $\mathbf{w}$ to exit a local minima. Broadly, this first exit time can be shown to only depend on the 'width' of the minima and the value of α. Imkeller and Pavlyukevich (2006) show that for the one-dimensional case the expected first exit time is:

$$\mathbb{E}[\tau] \propto \frac{\alpha}{2}\frac{a^\alpha}{b}, \tag{2.15}$$

where a is the width of the local minima, and $b = \rho^{\frac{\alpha-1}{\alpha}}\sigma$ is the noise scale. For decreasing α and noise scale b, under more heavy-tailed noise, SGD favours ever wider minima. Imkeller et al. (2010) generalise this result to the D-dimensional case and the result is broadly similar. Given a D-dimensional vector of tail parameters, the first exit time depends on the width in the dimension i where the Lévy process has the smallest α_i, i.e the direction in which it is most heavy-tailed.

More generally, we use this perspective on SGD to analyze the effect of GNIs on the dynamics of gradient descent in §4.

2.3 Noise Injections in Neural Networks

Consider an L layer network with no noise injections. We obtain the activations $\mathbf{h} \equiv \{\mathbf{h}_0, ..., \mathbf{h}_L\}$, where $\mathbf{h}_0 = \mathbf{x}$ is the input data. When performing a noise injection procedure, the value of the next layer's activations depend on the noised value (indicated by a tilde) of the previous layer. We denote the intermediate, soon-to-be-noised value of an activation,

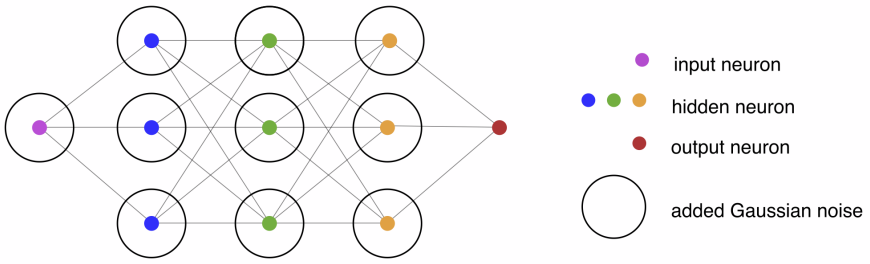


Figure 2.3: Here we show a fully connected network experiencing noise injections on its activations.

for a network consisting of dense layers (also known as a multi-layer perceptron: MLP):

$$\hat{\mathbf{h}}_i = \kappa(\mathbf{W}_i \tilde{\mathbf{h}}_{i-1}), \tag{2.16}$$

where κ is some non-linearity. This intermediate value is then itself noised by some element-wise operation $\circ$:

$$\tilde{\mathbf{h}}_i = \hat{\mathbf{h}}_i \circ \boldsymbol{\epsilon}_i. \tag{2.17}$$

We can, for example, add isotropic Gaussian noise with covariance $\sigma^2\mathbf{I}$, $\sigma \in \mathbb{R}$ to each hidden layer unit:

$$\tilde{\mathbf{h}}_i = \hat{\mathbf{h}}_i + \boldsymbol{\epsilon}_i, \;\; \boldsymbol{\epsilon}_i \sim \mathcal{N}(0, \sigma^2\mathbf{I}). \tag{2.18}$$

Under this procedure we have $p(\tilde{\mathbf{h}}_i|\hat{\mathbf{h}}_i, \boldsymbol{\epsilon}_i) = \mathcal{N}(\tilde{\mathbf{h}}_i|\hat{\mathbf{h}}_i, \sigma^2\mathbf{I})$. Alternatively, we can multiply by Gaussian noise, so $\tilde{\mathbf{h}}_i = \hat{\mathbf{h}}_i \odot \boldsymbol{\epsilon}_i, \;\; \boldsymbol{\epsilon}_i \sim \mathcal{N}(1, \sigma^2\mathbf{I})$ (where $\odot$ is element-wise multiplication). This is an injection generally referred to as *Gaussian Dropout* (Srivastava et al., 2014; Kingma et al., 2015; Wang and Manning, 2013). Under both forms of noise injection we obtain a noised loss $\tilde{\mathcal{L}}$, which corresponds to the original loss function $\mathcal{L}$ when experiencing GNI under the noising process $p(\tilde{\mathbf{h}}|\mathbf{h}) = \prod_i^L p(\tilde{\mathbf{h}}_i|\mathbf{h}_i)$. This process is obtained, as described above, by applying the perturbations $\boldsymbol{\epsilon}$ to the activations at each layer. We can describe the noised loss, when marginalising over the noising process (hence the lack

of dependence on $\boldsymbol{\epsilon}$), as:

$$\tilde{\mathcal{L}}(\mathcal{B}; \mathbf{w}) = \mathbb{E}_{\mathbf{x},\mathbf{y}\sim\mathcal{B}} \log \mathbb{E}_{\tilde{\mathbf{h}}\sim p(\tilde{\mathbf{h}}|\mathbf{h})} p_{\mathbf{w}}(\mathbf{y}|\tilde{\mathbf{h}}). \tag{2.19}$$

Generally one estimates this noised likelihood using Jensen's inequality:

$$\tilde{\mathcal{L}}(\mathcal{B}; \mathbf{w}) \geq \mathbb{E}_{\substack{\tilde{\mathbf{h}}\sim p(\tilde{\mathbf{h}}|\mathbf{h}) \\ \mathbf{x},\mathbf{y}\sim\mathcal{B}}} \log p_{\mathbf{w}}(\mathbf{y}|\tilde{\mathbf{h}}). \tag{2.20}$$

This lower bound means we have shifted probability mass away from the maximum likelihood estimate of the label $\mathbf{y}$ to other labels. We have effectively introduced greater uncertainty into our model: under the noising process detailed in equation (2.18) the expected value of the log likelihood of the model $p_{\mathbf{w}}$ shifts away from the true target $\mathbf{y}$. Intuitively, the error from Jensen's inequality decreases as our noise distribution standard deviation σ decreases. As $\sigma \to 0$, for an individual noise sample $\boldsymbol{\epsilon}$, $\mathcal{L}(\mathcal{B}; \mathbf{w}, \boldsymbol{\epsilon}) \to \mathcal{L}(\mathcal{B}; \mathbf{w})$ As such, σ controls the degree to which we regularise our marginal likelihood.

Recent work by Wei et al. (2020) studies the effect of Bernoulli multiplicative noise (simply referred to as *dropout*, where $\epsilon \sim \text{Ber}(p)$, p being the 'dropout rate'). They study this noise by taking a divide and conquer approach to their analysis. The first effect they study is the *explicit effect* of the injections, which is an added term to the loss function obtained when marginalising out the noise injections. When marginalising over the injected noise, we are effectively studying the 'mean' effect of the noise injections. They show that this term is positive, offering a *consistent objective* for neural networks to optimize, and has a regularising effect which improves generalisation performance of neural networks, though they don't offer an analysis as to why this is the case. They then study the implicit effect, which is the remainder of the terms marginalised out in the implicit effect. This corresponds to a study of the effect of the noisy loss' deviation from the 'mean' of the explicit effect. By its stochastic nature, this effect adds an additional term to the loss that is neither consistently positive or negative and as such requires other tools to study its effects on neural network. Wei et al. (2020) study this effect empirically for Bernoulli dropout but don't offer a theoretical framework by which to understand the mechanisms of this effect. As a formal summary, the batch-wise loss under noise injections

$\tilde{\mathcal{L}}$ can be expressed as the sum of the original non-noisy loss, to which we added the explicit and implicit effects:

$$\tilde{\mathcal{L}}(\mathcal{B}; \mathbf{w}, \boldsymbol{\epsilon}) = \mathcal{L}(\mathcal{B}; \mathbf{w}) + \underbrace{\mathbb{E}_{\boldsymbol{\epsilon}}\left[\Delta\tilde{\mathcal{L}}(\mathcal{B}; \mathbf{w}, \boldsymbol{\epsilon})\right]}_{\text{explicit effect: 'mean' effect of injections}} + \underbrace{\left(\Delta\tilde{\mathcal{L}}(\mathcal{B}; \mathbf{w}, \boldsymbol{\epsilon}) - \mathbb{E}_{\boldsymbol{\epsilon}}\left[\Delta\tilde{\mathcal{L}}(\mathcal{B}; \mathbf{w}, \boldsymbol{\epsilon})\right]\right)}_{\text{implicit effect: 'deviation' from mean effect}}, \tag{2.21}$$

where $\Delta\tilde{\mathcal{L}}(\mathcal{B}; \mathbf{w}, \boldsymbol{\epsilon}) = \tilde{\mathcal{L}}(\mathcal{B}; \mathbf{w}, \boldsymbol{\epsilon}) - \mathcal{L}(\mathcal{B}; \mathbf{w})$ is the difference between the noised loss and the original non-noisy loss. In our study of noise injections in §3 & §4 we take a similar approach and divide our study into that of the implicit and explicit effects.

It can also be helpful to study noise injections under the perspective of invariant models (Eaton, 1989; Nalisnick and Smyth, 2018; Cohen and Welling, 2016).

Definition 2.1 (q-invariance)**.** *A family of statistical models $p_{\mathbf{w}}$, indexed by their parameters $\mathbf{w}$, is said to be q-invariant if $p_{\mathbf{w}}(\mathbf{y}|\tilde{\mathbf{x}}) = p_{\mathbf{w}}(\mathbf{y}|\mathbf{x}), \forall\tilde{\mathbf{x}}$, where $\tilde{\mathbf{x}}$ is drawn from the perturbative process $q(\tilde{\mathbf{x}}|\mathbf{x})$.*

The key insight to glean from this definition and equations (2.19) and (2.20) is that we are extending this concept of invariance, beyond the data-domain, to the activation domain when using noise injections. We are training our model to keep predictions of the labels $\mathbf{y}$ constant under the perturbative process $p(\tilde{\mathbf{h}}|\mathbf{h})$. Similarly Bernoulli dropout (Srivastava et al., 2014), which multiplies samples from a Bernoulli distribution with parameter r to the activations of a network, can be interpreted as a perturbative process with rate r on the activations of a neural network:

$$p(\tilde{\mathbf{h}}|\mathbf{h}) = \begin{cases} 0, & \text{with probability } r \\ \mathbf{h}, & \text{with probability } 1 - r \end{cases}. \tag{2.22}$$

We are effectively training the model to be invariant to the perturbative process, whether it be samples from a Bernoulli or a Gaussian.

Note that the perturbative processes and noising operations can also be applied to the weights of a neural network. Though our work here takes the perspective of noised

activations, our results are directly translatable to the setting of perturbed weights.

Adversarial Robustness and Noise

The noise injection methods we have exposed and detailed previously operate in a manner akin to randomised smoothing techniques for deterministic classifiers (Cohen et al., 2019; Lecuyer et al., 2019). Whereas noise injections used for regularising purposes tend to be inflicted at *training time*, randomized smoothing techniques operate at test time, and add noise to the inputs of neural networks when making new predictions; leveraging the randomness of noise injections to obfuscate *adversarial attacks*.

In adversarial settings for classification, a hypothetical adversary looks to apply small distortions to data that produce network predictions that differ from the 'true' data label. In the generative model setting, particularly in the case of Variational Autoencoders (VAEs, which we introduce in full in the next section), the adversary aims to produce reconstructions from the VAE decoder (more on this in §5) that are close to a target image (Gondim-Ribeiro et al., 2018a,b).

Critically, randomized smoothing techniques protect against adversarial attacks by increasing the margin of robustness of neural networks. Formally this margin is the smallest spherical radius r around a datapoint x, for which a classifier outputs the same prediction. Cohen et al. (2019) show that this margin scales linearly with the standard deviation of the injected noise (when using Gaussian noise). We use this idea of a minimum margin of robustness when developing the theory on the adversarial robustness of Variational Autoencoders in §5.

2.3.1 Deep Latent Variable Models

In probabilistic models, latent variables refer to hidden variables for which we cannot obtain direct measurements. We can define a factorisation of the joint distribution for each pair of datapoints and latents, and a set of associated model parameters. In the Bayesian framework, we can define priors over variables expressing prior knowledge of the distribution of variables, and different functional forms for probability distributions; the choice of which as a whole define a *generative model*. We can even leverage neural

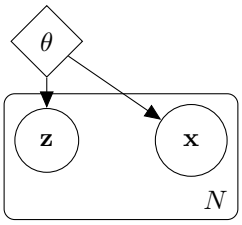


Figure 2.4: Graph defined by Equation (2.23). Diamonds represent deterministic variables and circles represent random variables. Arrows indicate conditional independence between two variables. The plate represents the N pairs of x, z variables.

networks to parameterise the functional forms of these distributions, forming Deep Latent Variable Models (DLVMs).

As an example, consider the case where we view a set of model parameters $\theta \in \Theta$ as being deterministic, and not as a random variable to which we assign a prior (as a proper Bayesian model would do). For issues of scalability and tractability this is a perspective that is often taken by DLVMs. We have a dataset with N points $\mathcal{D} = \{\mathbf{x}_i\}$ $(i = 1 \dots N,\ \mathbf{x}_i \in \mathcal{X})$, and a set of associated latents $\mathbf{Z} = \{\mathbf{z}_i\}$, $(i = 1 \dots N,\ \mathbf{z}_i \in \mathcal{Z})$. The joint over data and latents is then given by,

$$p_\theta(\mathcal{D}; \mathbf{Z}) = p_\theta(\mathcal{D}|\mathbf{Z})p_\theta(\mathbf{Z}), \theta \in \Theta. \tag{2.23}$$

As before, in the case of supervised learning for neural networks, our aim is to find the maximum likelihood estimate for our parameters θ:

$$\theta^* = \arg\max_{\theta \in \Theta} p_\theta(\mathcal{D}) = \int d\mathbf{Z} p_\theta(\mathcal{D}|\mathbf{Z})p_\theta(\mathbf{Z}). \tag{2.24}$$

We may also wish to learn the posterior of our latents given data. By Bayes' Rule this is:

$$p_\theta(\mathbf{Z}|\mathcal{D}) = \frac{\prod_i^N p_\theta(\mathbf{x}_i|\mathbf{z}_i)p_\theta(z_i)}{p_\theta(\mathcal{D})}, \tag{2.25}$$

where $p_\theta(\mathcal{D})$ is given by the previous equation.

Both the posterior and the maximum likelihood estimate of parameters involve *often intractable* integrals over the latents $\mathbf{Z}$ contained in the evidence $p_\theta(\mathcal{D})$. The development of methods that can both approximate this integral and scale to large datasets has been

at the heart of the development of DLVMs. One such method, amongst others, has been variational inference; where instead of calculating the evidence of data $p_\theta(\mathcal{D})$ directly, we calculate a lower bound on it, the *evidence lower bound*.

Variational Bayesian Methods

Variational Bayesian methods are a family of techniques for performing inference on intractable integrals, such as calculating the posterior over latent variables $\mathbf{Z}$ given data $\mathbf{x} \in \mathcal{X}$. The posterior distribution over this set of unobserved variables is approximated by a variational distribution q_ϕ: $p_\theta(\mathbf{Z}|\mathcal{D}) \approx q_\phi(\mathbf{Z})$, where q is chosen to belong to a simpler family of distributions $\mathcal{Q}$ than $p_\theta(\mathbf{Z}|\mathcal{D})$ (Blei et al., 2017), and ϕ is another set of parameters chosen to be deterministic. In Variational Inference the aim is then to minimize the Kullback-Leibler (KL) divergence between $q_\phi(\mathbf{Z})$ and $p_\theta(\mathbf{Z}|\mathcal{D})$ (Kullback and Leibler, 1951; Blei et al., 2017):

$$q_\phi^*(\mathbf{Z}) = \underset{q_\phi^*(\mathbf{Z})\in\mathcal{Q}}{\arg\min}\, \mathrm{KL}(q_\phi(\mathbf{Z})||p_\theta(\mathbf{Z}|\mathcal{D})). \tag{2.26}$$

This KL is equivalent to:

$$\mathrm{KL}(q_\phi(\mathbf{Z})||p_\theta(\mathbf{Z}|\mathcal{D})) = -\mathbb{E}_{q_\phi(\mathbf{Z})}[\log p_\theta(\mathcal{D}|\mathbf{Z})] + \mathrm{KL}(q_\phi(\mathbf{Z})||p_\theta(\mathbf{Z})) + \log p_\theta(\mathcal{D}) \tag{2.27}$$

$$= -\mathcal{L}(\mathcal{D};\theta,\phi) + \log p_\theta(\mathcal{D}), \tag{2.28}$$

where $\mathcal{L}(\mathcal{D};\theta,\phi)$ is the evidence lower bound (ELBO) of the data, and $p_\theta(\mathbf{Z})$ is typically some simple factorisable prior over latents, such as an isotropic Gaussian. It can be shown using Jensen's inequality that $\mathcal{L}(\mathcal{D};\theta,\phi) \leq \log p_\theta(\mathcal{D})$, hence the name 'evidence lower bound'.

Minimising this KL divergence w.r.t to ϕ is equivalent to maximising the ELBO of the data $\mathcal{L}(\mathcal{D};\theta,\phi)$ over ϕ, given the current estimate for θ. As this KL decreases, so too does the discrepancy between the ELBO and the evidence $\log p_\theta(\mathcal{D})$, and our approximation of the 'true' evidence improves. If we chose the prior and the variational posterior to be of simple families for which the KL is a tractable object (such as conjugate exponentials), we now have a much simpler objective to compute.

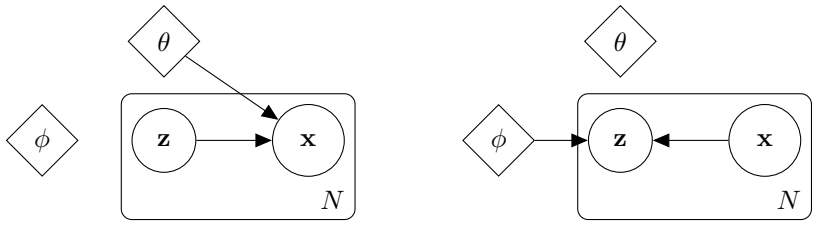


Figure 2.5: VAE DAG: the generative model **[left]** and variational posterior **[right]**. Symbols are the same as in Figure 2.4.

Even so, our aim ultimately is to scale these methods to extremely large datasets. In its current form, the ELBO does not offer gradients through which to differentiate for gradient descent based algorithms; which can easily scale to millions of datapoints. The innovation behind DLVMs is then to re-express the latents in such a manner so as to be differentiable w.r.t the variational posterior parameters ϕ. To illustrate this, and because they form the bulk of our work on DLVMs in §5 & §6, we consider the example of Variational Autoencoders (VAEs).

Variational Autoencoders

VAEs, and models they have inspired, are deep neural networks that can perform variational inference for high dimensional data and large datasets. They introduce a joint distribution over data $\mathbf{x}$ (in data-space $\mathcal{X}$) and a set of latent variables $\mathbf{z}$ (in latent-space $\mathcal{Z}$): $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})$ where $p_\theta(\mathbf{x}|\mathbf{z})$ is a distribution that matches the properties of the data. The parameters of the likelihood $p_\theta(\mathbf{x}|\mathbf{z})$ are encoded by neural networks which take in samples of $\mathbf{z}$. A common choice of prior for the space $\mathcal{Z}$ is the unit normal Gaussian: $p(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$. As mentioned before, exact inference for these models is intractable, and one performs *amortised* stochastic variational inference by introducing another inference neural network for the latent variables, $q_\phi(\mathbf{z}|\mathbf{x})$, which often encodes the parameters of another Gaussian, $\mathcal{N}(\mu_\phi(\mathbf{x}), \Sigma_\phi(\mathbf{x}))$ (Kingma et al., 2015).

In the context of VAEs, we use *amortised* Variational Inference, which refers to the fact that the parameters of the distributions $p_\theta(\mathbf{x}|\mathbf{z})$ and $q_\phi(\mathbf{z}|\mathbf{x})$ are data-dependent, meaning they vary per-datapoint $\mathbf{x}$. This amortisation allows for variational inference to scale to very large datasets.

Given the factorisation of the joint specified above and given the amortisation of in-

ference we can write an objective (Kingma and Welling, 2014):

$$\mathcal{L}(\mathcal{D};\theta,\phi) = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}\left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))\right]. \tag{2.29}$$

q_ϕ is often referred to as the encoder network and p_θ is referred to as the decoder network. Typically one could not propagate gradients through the parameterised distributions to train the encoder and decoder neural networks via gradient descent. To be able to do so we employ the reparameterisation trick, which re-expresses the latent variables of the VAE in deterministic form $\mathbf{z} = g(\boldsymbol{\epsilon}, \phi)$, such that $\mathbf{z}$ is then differentiable w.r.t the parameters ϕ. We must thus be able to find a distribution $p(\boldsymbol{\epsilon})$, such that

$$\mathbf{z} = g(\boldsymbol{\epsilon}, \phi), \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon}) \iff \mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}). \tag{2.30}$$

For Gaussian variational posteriors for example we can draw samples from $q_\phi(\mathbf{z}|\mathbf{x})$ as:

$$\mu_\phi(\mathbf{x}) + (\Sigma_\phi(\mathbf{x}))^{\frac{1}{2}}\,\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{2.31}$$

where $\mu_\phi(\mathbf{x})$ and $\Sigma_\phi(\mathbf{x}))$ are the mean and covariance parameterised by the encoder. This is clearly equivalent to having sampled from the variational posterior $\mathcal{N}(\mu_\phi(\mathbf{x}), \Sigma_\phi(\mathbf{x}))$, and this expression is differentiable w.r.t to the encoder networks $\mu_\phi(\mathbf{x})$ and $\Sigma_\phi(\mathbf{x}))$ as $q_\phi(\mathbf{z}|\mathbf{x})$ is now deterministic w.r.t ϕ. We coin this layer the 'stochastic' layer of the VAE. **Note that it is clearly analogous to a Gaussian noise addition to** $\mu_\phi(\mathbf{x})$. We use this observation at length in §5 & §6.

We then approximate the likelihood $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right]$ using M Monte-Carlo samples from this stochastic layer:

$$\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] \approx \frac{1}{M}\sum_{i=1}^{M} \log p_\theta(\mathbf{x}|\mu_\phi(\mathbf{x}) + (\Sigma_\phi(\mathbf{x}))^{\frac{1}{2}}\,\boldsymbol{\epsilon}_i), \qquad \boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

Draws from $p_\theta(\mathbf{x}|\mathbf{z})$ are termed 'reconstructions' of the original data $\mathbf{x}$ in the VAE literature. These samples from the variational posterior can also be used to approximate the KL divergence, but it has a tractable form in the case of a Gaussian prior and variational posterior.

As we can use gradient descent based methods to train VAEs, we can also use stochastic batching for training as in 3.2.1. Our training objective then becomes:

$$\mathcal{L}(\mathcal{B};\theta,\phi) = \mathbb{E}_{\mathbf{x}\sim\mathcal{B}}\left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))\right], \tag{2.32}$$

where we have substituted our expectation over the dataset for an expectation over a sampled mini-batch $\mathcal{B}$.

CHAPTER 3

EXPLICIT REGULARISATION IN GAUSSIAN NOISE INJECTIONS

This work was written in collaboration with Matthew Willetts, Umut Şimşekli, Stephen Roberts and Chris Holmes.

3.1 Introduction 27

3.2 Background 28

3.2.1 Gaussian Noise Injections 28

3.2.2 Sobolev Spaces 29

3.3 The Explicit Effect of Gaussian Noise Injections 31

3.4 Fourier Domain Regularisation 35

3.5 Related Work 41

3.6 Conclusion 41

Abstract

We study the regularisation induced in neural networks by Gaussian noise injections (GNIs). Though such injections have been extensively studied when applied to data, there have been few studies on understanding the regularising effect they induce when applied to network activations. Here we derive the explicit regulariser of GNIs, obtained by marginalising out the injected noise, and show that it penalises functions with high-frequency components in the Fourier domain; particularly in layers closer to a neural network's output. We show analytically and empirically that such regularisation produces calibrated classifiers with large classification margins.

3.1 Introduction

Noise injections are a family of methods that involve adding or multiplying samples from a noise distribution, typically an isotropic Gaussian, to the weights or activations of a neural network during training. The benefits of such methods are well documented. Models trained with noise often generalise better to unseen data and are less prone to overfitting (Srivastava et al., 2014; Kingma et al., 2015; Poole et al., 2014).

Even though the regularisation conferred by Gaussian noise injections (GNIs) can be observed empirically, and the benefits of noising data are well understood theoretically (Bishop, 1995; Cohen et al., 2019; Webb, 1994), there have been few studies on understanding the benefits of methods that inject noise *throughout* a network. Here we study the *explicit* regularisation of such injections, which is a positive term added to the loss function obtained when we marginalise out the noise we have injected.

Concretely our contributions are:

- We derive an analytic form for an explicit regulariser that explains most of GNIs' regularising effect.
- We show that this regulariser penalises networks that learn functions with high-frequency content in the Fourier domain and most heavily regularises neural network layers that are closer to the output. See Figure 3.1 for an illustration.
- Finally, we show analytically and empirically that this regularisation induces larger

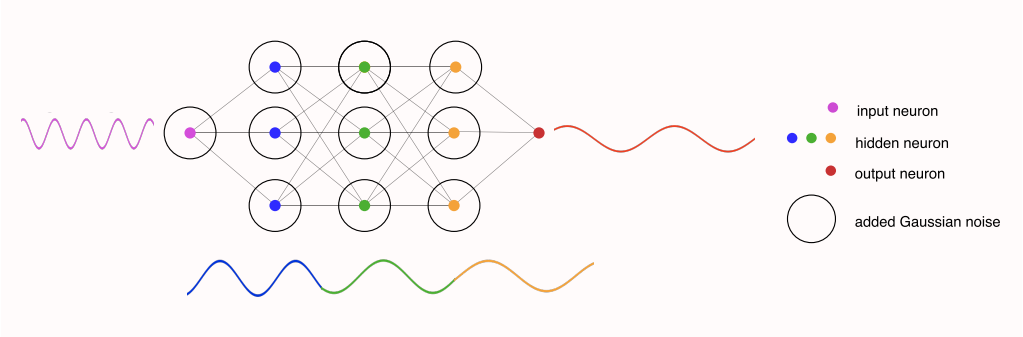


Figure 3.1: Here we illustrate the effect of GNIs injected throughout a network's activations. Each coloured dot represents a neuron's activations. We add GNIs, represented as circles, to each layer's activations bar the output layer. GNIs induce a network for which each layer learns a progressively lower frequency function, represented as a sinusoid matching in colour to its corresponding layer.

classification margins and better calibration of models.

3.2 Background

3.2.1 Gaussian Noise Injections

Training a neural network involves optimising network parameters to maximise the marginal likelihood of a set of labels given features via gradient descent. With a training dataset $\mathcal{D}$ composed of N data-label pairs of the form $(\mathbf{x}, \mathbf{y})$ $\mathbf{x} \in \mathbb{R}^d, \mathbf{y} \in \mathbb{R}^m$ and a feed-forward neural network with M parameters divided into L layers: $\mathbf{w} \equiv \{\mathbf{W}_1, ..., \mathbf{W}_L\}$, $\mathbf{w} \in \mathbb{R}^M$, our objective is to minimise the expected negative log likelihood of labels $\mathbf{y}$ given data $\mathbf{x}$, $-\log p_{\mathbf{w}}(\mathbf{y}|\mathbf{x})$, and find the optimal set of parameters $\mathbf{w}^*$ satisfying:

$$\mathbf{w}_* = \arg\min_{\mathbf{w}} \mathcal{L}(\mathcal{D}; \mathbf{w}), \qquad \mathcal{L}(\mathcal{D}; \mathbf{w}) := -\mathbb{E}_{\mathbf{x},\mathbf{y}\sim\mathcal{D}} \left[\log p_{\mathbf{w}}(\mathbf{y}|\mathbf{x})\right] . \tag{3.1}$$

Under stochastic optimisation algorithms, such as Stochastic Gradient Descent (SGD), we estimate $\mathcal{L}$ by sampling a mini-batch of data-label pairs $\mathcal{B} \subset \mathcal{D}$.

$$\mathcal{L}(\mathcal{B}; \mathbf{w}) = -\mathbb{E}_{\mathbf{x},\mathbf{y}\sim\mathcal{B}} \log p_{\mathbf{w}}(\mathbf{y}|\mathbf{x}) \approx \mathcal{L}(\mathcal{D}; \mathbf{w}). \tag{3.2}$$

Consider an L layer network with no noise injections and a non-linearity κ at each layer. We obtain the activations $\mathbf{h} = \{\mathbf{h}_0, ..., \mathbf{h}_L\}$, where $\mathbf{h}_0 = \mathbf{x}$ is the input data *before* any

noise is injected. For a network consisting of dense layers (a.k.a. a multi-layer perceptron: MLP) we have that:

$$\mathbf{h}_k(\mathbf{x}) = \kappa(\mathbf{W}_k \mathbf{h}_{k-1}(\mathbf{x})). \tag{3.3}$$

What happens to these activations when we inject noise? First, let $\boldsymbol{\epsilon}$ be the set of noise injections at each layer: $\boldsymbol{\epsilon} = \{\boldsymbol{\epsilon}_0, ..., \boldsymbol{\epsilon}_{L-1}\}$. When performing a noise injection procedure, the value of the next layer's activations depends on the noised value of the previous layer. We denote the intermediate, soon-to-be-noised value of an activation as $\widehat{\mathbf{h}}_k$ and the subsequently noised value as $\widetilde{\mathbf{h}}_k$:

$$\widehat{\mathbf{h}}_k(\mathbf{x}) = \kappa\left(\mathbf{W}_k \widetilde{\mathbf{h}}_{k-1}(\mathbf{x})\right), \qquad \widetilde{\mathbf{h}}_k(\mathbf{x}) = \widehat{\mathbf{h}}_k(\mathbf{x}) \circ \boldsymbol{\epsilon}_k, \tag{3.4}$$

where $\circ$ is some element-wise operation. We can, for example, add or multiply Gaussian noise to each hidden layer unit. In the additive case, we obtain:

$$\widetilde{\mathbf{h}}_k(\mathbf{x}) = \widehat{\mathbf{h}}_k(\mathbf{x}) + \boldsymbol{\epsilon}_k, \qquad \boldsymbol{\epsilon}_k \sim \mathcal{N}(0, \sigma_k^2 \mathbf{I}). \tag{3.5}$$

The multiplicative case can be rewritten as an activation-scaled addition:

$$\widetilde{\mathbf{h}}_k(\mathbf{x}) = \widehat{\mathbf{h}}_k(\mathbf{x}) + \boldsymbol{\epsilon}_k, \qquad \boldsymbol{\epsilon}_k \sim \mathcal{N}\left(0, \widehat{\mathbf{h}}_k^2(\mathbf{x}) \sigma_k^2 \mathbf{I}\right). \tag{3.6}$$

Here we focus our analysis on noise *additions*, but through equation (3.6) we can translate our results to the multiplicative case.

3.2.2 Sobolev Spaces

To define a Sobolev Space we use the generalisation of the derivative for multivariate functions of the form $g : \mathbb{R}^d \to \mathbb{R}$. We use a multi-index notation $\alpha \in \mathbb{R}^d$ which defines mixed partial derivatives. We denote the α^{th} derivative of g with respect to its input $\mathbf{x}$ as $D^\alpha g(\mathbf{x})$.

$$D^\alpha g = \frac{\partial^{|\alpha|} g}{\partial x_1^{\alpha_1} \dots \partial x_d^{\alpha_d}}$$

where $|\alpha| = \sum_{i=1}^{d} |\alpha_i|$. Note that $\mathbf{x}^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \dots x_d^{\alpha_d}$ and $\alpha! = \alpha_1! \alpha_2 \dots \alpha_d!$.

Definition 3.1 (Cucker and Smale (2002)). *Sobolev spaces are denoted $W^{l,p}(\Omega), \Omega \subset \mathbb{R}^d$, where l, the order of the space, is a non-negative integer and $p \geq 1$. The Sobolev space of index (l,p) is the space of locally integrable functions $f : \Omega \to \mathbb{R}$ such that for every multi-index α where $|\alpha| < l$ the derivative $D^\alpha f$ exists and $D^\alpha f \in L^p(\Omega)$. The norm in such a space is given by $\|f\|_{W^{l,p}(\Omega)} = \left(\sum_{|\alpha| \leq l} \int_\Omega |D^\alpha f(\mathbf{x})|^p d\mathbf{x} \right)^{\frac{1}{p}}$.*

For $p = 2$ these spaces are Hilbert spaces, with a dot product that defines the L_2 norm of a function's derivatives. Further these Sobolev spaces can be defined in a measure space with *finite* measure μ. We call such spaces finite measure spaces of the form $W_\mu^{l,p}(\mathbb{R}^d)$ and these are the spaces of locally integrable functions such that for every α where $|\alpha| < l$, $D^\alpha f \in L_\mu^p(\mathbb{R}^d)$, the L^p space equipped with the measure μ. The norm in such a space is given by (Hornik, 1991):

$$\|f\|_{W_\mu^{l,p}(\mathbb{R}^d)} = \left(\sum_{|\alpha| \leq l} \int_{\mathbb{R}^d} |D^\alpha f(\mathbf{x})|^p d\mu(\mathbf{x}) \right)^{\frac{1}{p}}, f \in W_\mu^{l,p}(\mathbb{R}^d), |\mu(\mathbf{x})| < \infty \ \forall \mathbf{x} \in \mathbb{R}^d \quad (3.7)$$

Generally a Sobolev space over a compact subset Ω of $\mathbb{R}^d$ can be expressed as a weighted Sobolev space with a measure μ which has compact support on Ω (Hornik, 1991).

Hornik (1991) have shown that neural networks with continuous activations, which have continuous and bounded derivatives up to order l, such as the sigmoid function, are universal approximators in the *weighted* Sobolev spaces of order l, meaning that they form a dense subset of Sobolev spaces. Further, Czarnecki et al. (2017) have shown that networks that use piecewise linear activation functions (such as ReLU and its extensions) are *also* universal approximators in the Sobolev spaces of order 1 where the domain Ω is some compact subset of $\mathbb{R}^d$. As mentioned above, this is equivalent to being dense in a weighted Sobolev space on $\mathbb{R}^d$ where the measure μ has compact support. Hence, we can view a neural network, with sigmoid or piecewise linear activations to be a parameter that indexes a function in a weighted Sobolev space with index $(1,2)$, i.e. $f_\mathbf{w} \in W_\mu^{1,2}(\mathbb{R}^d)$.

3.3 The Explicit Effect of Gaussian Noise Injections

Here we consider the case where we noise all layers with *isotropic* noise, except the final predictive layer which we also consider to have no activation function. We can express the effect of the Gaussian noise injection on the cost function as an added term $\Delta\mathcal{L}$, which is dependent on $\boldsymbol{\mathcal{E}}_L$, the noise accumulated on the final layer L from the noise additions $\boldsymbol{\epsilon}$ on the previous hidden layer activations.

$$\widetilde{\mathcal{L}}(\mathcal{B}; \mathbf{w}, \boldsymbol{\epsilon}) = \mathcal{L}(\mathcal{B}; \mathbf{w}) + \Delta\mathcal{L}(\mathcal{B}; \mathbf{w}, \boldsymbol{\mathcal{E}}_L) \tag{3.8}$$

To understand the regularisation induced by GNIs, we want to study the regularisation that these injections induce *consistently* from batch to batch. To do so, we want to remove the stochastic component of the GNI regularisation and extract a regulariser that is of consistent sign. Regularisers that change sign from batch-to-batch do not give a constant objective to optimise, making them unfit as regularisers (Botev et al., 2017; Sagun et al., 2018; Wei et al., 2020).

As such, we study the explicit regularisation these injections induce by way of the expected regulariser, $\mathbb{E}_{\boldsymbol{\epsilon}\sim p(\boldsymbol{\epsilon})}\left[\Delta\mathcal{L}(\cdot)\right]$ that marginalises out the injected noise $\boldsymbol{\epsilon}$. To lighten notation, we denote this as $\mathbb{E}_{\boldsymbol{\epsilon}}\left[\Delta\mathcal{L}(\cdot)\right]$. We extract R, a constituent term of the expected regulariser that dominates the remainder terms in norm, and is *consistently positive*.

Because of these properties, R provides a lens through which to study the effect of GNIs. As we show, this term has a connection to the Sobolev norm and the Fourier transform of the function parameterised by the neural network. Using these connections we make inroads into better understanding the regularising effect of noise injections on neural networks.

To begin deriving this term, we first need to define the accumulated noise $\boldsymbol{\mathcal{E}}_L$. We do so by applying a Taylor expansion to each noised layer. As in Section 3.2.2 we use the generalisation of the derivative for multivariate functions using a multi-index α. For example $D^\alpha h_{k,i}(\mathbf{h}_{k-1}(\mathbf{x}))$ denotes the α^{th} derivative of the i^{th} activation of the k^{th} layer $(h_{k,i})$ with respect to the preceding layer's activations $\mathbf{h}_{k-1}(\mathbf{x})$ and $D^\alpha\mathcal{L}(\mathbf{h}_k(\mathbf{x}), \mathbf{y})$ denotes the α^{th} derivative of the loss with respect to the non-noised activations $\mathbf{h}_k(\mathbf{x})$.

Proposition 3.1. *Consider an L layer neural network experiencing isotropic GNIs at each layer $k \in [0, \dots, L-1]$ of dimensionality d_k. We denote this added noise as $\boldsymbol{\epsilon} = \{\boldsymbol{\epsilon}_0, ..., \boldsymbol{\epsilon}_{L-1}\}$. We assume $\mathbf{h}_L$ is in C^∞ the class of infinitely differentiable functions. We can define the accumulated noise at each layer k using a multi-index $\alpha_k \in \mathbb{N}^{d_{k-1}}$:*

$$\mathcal{E}_{L,i} = \sum_{|\alpha_L|=1}^{\infty} \frac{1}{\alpha_L!} \left(D^{\alpha_L} h_{L,i}(\mathbf{h}_{L-1}(\mathbf{x}))\right) \boldsymbol{\mathcal{E}}_{L-1}^{\alpha_L}, \ i = 1, \dots, d_L$$

$$\mathcal{E}_{k,i} = \epsilon_{k,i} + \sum_{|\alpha_k|=1}^{\infty} \frac{1}{\alpha_k!} \left(D^{\alpha_k} h_{k,i}(\mathbf{h}_{k-1}(\mathbf{x}))\right) \boldsymbol{\mathcal{E}}_{k-1}^{\alpha_k}, \ i = 1, \dots, d_k, \ k = 1 \dots L-1$$

$$\boldsymbol{\mathcal{E}}_0 = \boldsymbol{\epsilon}_0$$

where $\mathbf{x}$ is drawn from the dataset $\mathcal{D}$, $\mathbf{h}_k$ are the activations before any noise is added, as defined in Equation (3.3).

See Appendix A.1.1 for the proof. Given this form for the accumulated noise, we can now define the expected regulariser. For compactness of notation, we denote each layer's Jacobian as $\mathbf{J}_k \in \mathbb{R}^{d_L \times d_k}$ and the Hessian of the loss with respect to the final layer as $\mathbf{H}_L \in \mathbb{R}^{d_L \times d_L}$. Each entry of $\mathbf{J}_k$ is a partial derivative of $f_{k,i}^\theta$, the function from layer k to the i^{th} network output, $i = 1...d_L$.

$$\mathbf{J}_k(\mathbf{x}) = \begin{bmatrix} \frac{f_{k,1}^\theta}{\partial h_{k,1}} & \frac{f_{k,1}^\theta}{\partial h_{k,2}} & \cdots \\ \vdots & \ddots & \\ \frac{f_{k,d_L}^\theta}{\partial h_{k,1}} & & \frac{f_{k,d_L}^\theta}{\partial h_{k,d_k}} \end{bmatrix}, \ \mathbf{H}_L(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \frac{\partial^2 \mathcal{L}}{\partial h_{L,1}^2} & \frac{\partial^2 \mathcal{L}}{\partial h_{L,1} \partial h_{L,2}} & \cdots \\ \vdots & \ddots & \\ \frac{\partial^2 \mathcal{L}}{\partial h_{L,d_L} \partial h_{L,1}} & & \frac{\partial^2 \mathcal{L}}{\partial h_{L,d_L}^2} \end{bmatrix}$$

Using these notations we can now define the explicit regularisation induced by GNIs.

Theorem 3.2. *Consider an L layer neural network experiencing isotropic GNIs at each layer $k \in [0, \dots, L-1]$ of dimensionality d_k. We denote this added noise as $\boldsymbol{\epsilon} = \{\boldsymbol{\epsilon}_0, ..., \boldsymbol{\epsilon}_{L-1}\}$. We assume $\mathcal{L}$ is in C^∞ the class of infinitely differentiable functions. We can marginalise out the injected noise $\boldsymbol{\epsilon}$ to obtain an added regulariser:*

$$\mathbb{E}_{\boldsymbol{\epsilon}} \left[\Delta \mathcal{L}(\mathcal{B}; \mathbf{w}, \boldsymbol{\mathcal{E}}_L)\right] = \mathbb{E}_{(\mathbf{x},\mathbf{y}) \sim \mathcal{B}} \left[\frac{1}{2} \sum_{k=0}^{L-1} \left[\sigma_k^2 \mathrm{Tr} \left(\mathbf{J}_k^\intercal(\mathbf{x}) \mathbf{H}_L(\mathbf{x}, \mathbf{y}) \mathbf{J}_k(\mathbf{x}) \right) \right] \right] + \mathbb{E}_{\boldsymbol{\epsilon}} \left[\mathcal{C}(\mathcal{B}, \boldsymbol{\epsilon})\right]$$

where $\mathbf{h}_k$ are the activations before any noise is added, as in equation (3.3). $\mathbb{E}_{\boldsymbol{\epsilon}} [\mathcal{C}(\cdot)]$ is a

remainder term in higher order derivatives.

See Appendix A.1.2 for the proof and for the exact form of the remainder.

We denote the first term in Theorem 3.2 as:

$$R(\mathcal{B};\mathbf{w}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{B}}\left[\frac{1}{2}\sum_{k=0}^{L-1}\left[\sigma_k^2 \mathrm{Tr}\left(\mathbf{J}_k^\intercal(\mathbf{x})\mathbf{H}_L(\mathbf{x},\mathbf{y})\mathbf{J}_k(\mathbf{x})\right)\right]\right] \tag{3.9}$$

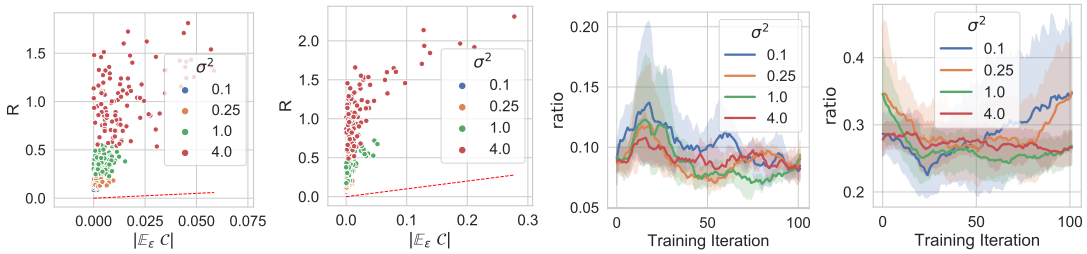
To understand the main contributors behind the regularising effect of GNIs, we first want to establish the relative importance of the two terms that constitute the explicit effect. We know that R is the added regulariser for the linearised version of a neural network, defined by its Jacobian. This linearisation well approximates neural network behaviour for *sufficiently wide* networks (Jacot et al., 2018; Chizat et al., 2019; Arora et al., 2019) in early stages of training (Chen et al., 2020), and we can expect R to dominate the remainder term in norm, which consists of higher order derivatives. In Figure 3.2 we show that this is the case for a range of GNI variances, datasets, and activation functions for networks with 256 neurons per layer; where the remainder is estimated as:

$$\mathbb{E}_{\boldsymbol{\epsilon}}\left[\mathcal{C}(\mathcal{B},\boldsymbol{\epsilon})\right] \approx \frac{1}{1000}\sum_{i=0}^{1000}\widetilde{\mathcal{L}}(\mathcal{B};\mathbf{w},\boldsymbol{\epsilon}_i) - R(\mathcal{B};\mathbf{w}) - \mathcal{L}(\mathcal{B};\mathbf{w}).$$

These results show that R is a significant component of the regularising effect of GNIs. It dominates the remainder $\mathbb{E}_{\boldsymbol{\epsilon}}\left[\mathcal{C}(\cdot)\right]$ in norm and is always positive, as we will show, thus offering a consistent objective for SGD to minimise. Given that R is a likely candidate for understanding the effect of GNIs; we further study this term separately in regression and classification settings.

Regularisation in Regression In the case of regression one of the most commonly used loss functions is the mean-squared error (MSE), which is defined for a data label pair $(\mathbf{x},\mathbf{y})$ as:

$$\mathcal{L}(\mathbf{x},\mathbf{y}) = \frac{1}{2}(\mathbf{y}-\mathbf{h}_L(\mathbf{x}))^2. \tag{3.10}$$



(a) BHP Sigmoid (b) CIFAR10 ELU (c) BHP Sigmoid (d) CIFAR10 ELU

Figure 3.2: In (a,b) we plot $R(\cdot)$ vs $\mathbb{E}_{\boldsymbol{\epsilon}}\left[\mathcal{C}(\cdot)\right]$ at initialisation for 6-layer-MLPs with GNIs at each 256-neuron layer with the same variance $\sigma^2 \in [0.1, 0.25, 1.0, 4.0]$ at each layer. Each point corresponds to one of 250 different network initialisation acting on a batch of size 32 for the classification dataset CIFAR10 and regression dataset Boston House Prices (BHP) datasets. The dotted red line corresponds to $y = x$ and demonstrates that for all batches and GNI variances R is greater than $\mathbb{E}_{\boldsymbol{\epsilon}}\left[\mathcal{C}(\cdot)\right]$. In (c,d) we plot ratio $= |\,\mathbb{E}_{\boldsymbol{\epsilon}}\left[\mathcal{C}(\cdot)\right]|/R(\cdot)$ in the first 100 training iteration for 10 randomly initialised networks. Shading corresponds to the standard deviation of values over the 10 networks. $R(\cdot)$ remains dominant in early stages of training as the ratio is less than 1 for all steps.

For this loss, the Hessians in Theorem 3.2 are simply the identity matrix. The explicit regularisation term, guaranteed to be positive is:

$$R(\mathcal{B}; \mathbf{w}) = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim\mathcal{B}}\left[\sum_{k=0}^{L-1} \sigma_k^2(\|\mathbf{J}_k(\mathbf{x})\|_F^2)\right]. \tag{3.11}$$

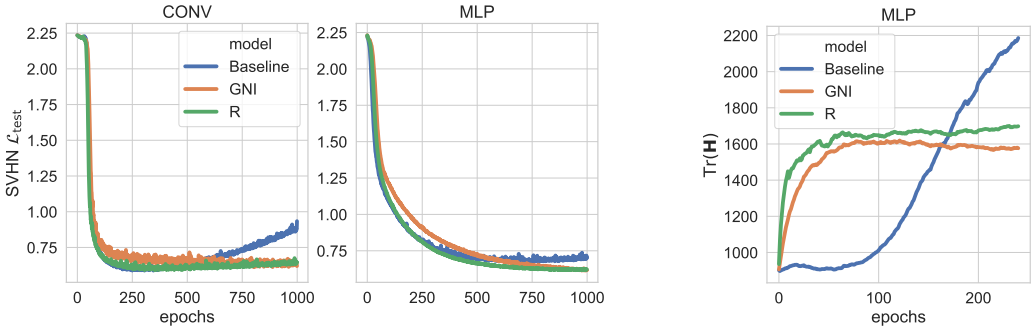
where σ_k^2 is the variance of the noise $\boldsymbol{\epsilon}_k$ injected at layer k and $\|\cdot\|_F$ is the Frobenius norm. See Appendix A.1.4 for a proof.

Regularisation in Classification In the case of classification, we consider the case of a cross-entropy (CE) loss. Recall that we consider our network outputs $\mathbf{h}_L$ to be the pre-softmax of the logits of the final layer. We denote $\mathbf{p}(\mathbf{x}) = \text{softmax}(\mathbf{h}_L(\mathbf{x}))$. For a pair $(\mathbf{x}, \mathbf{y})$ we have:

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = -\sum_{c=0}^{C} \mathbf{y}_c \log(\mathbf{p}(\mathbf{x}))_c), \tag{3.12}$$

where c indexes over C possible classes. The hessian $\mathbf{H}_L(\cdot)$ no longer depends on $\mathbf{y}$:

$$\mathbf{H}_L(\mathbf{x})_{i,j} = \begin{cases} \mathbf{p}(\mathbf{x})_i(1 - \mathbf{p}(\mathbf{x})_j) & i = j \\ -\mathbf{p}(\mathbf{x})_i\mathbf{p}(\mathbf{x})_j & i \neq j \end{cases} \tag{3.13}$$



(a) SVHN ELU (b) SVHN MLP Tr($\mathbf{H}$)

Figure 3.3: Figure (a) shows the test set loss for convolutional models (CONV) and 4 layer MLPs trained on SVHN with $R(\cdot)$ and GNIs for $\sigma^2 = 0.1$, and no noise (Baseline). Figure (b) shows the trace of the network parameter Hessian for a 2-layer, 32-unit-per-layer MLP where $\mathbf{H}_{i,j} = \frac{\partial \mathcal{L}}{\partial w_i \partial w_j}$, which is a proxy for the parameters' location in the loss landscape. All networks use ELU activations. See Appendix A.6 for more such results on other datasets and network architectures.

This Hessian is positive-semi-definite and $R(\cdot)$, guaranteed to be positive, can be written as:

$$R(\mathcal{B}; \mathbf{w}) = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim\mathcal{B}}\left[\sum_{k=0}^{L-1}\sigma_k^2 \sum_{i,j}(\text{diag}(\mathbf{H}_L(\mathbf{x}))^\intercal \mathbf{J}_k^2(\mathbf{x}))_{i,j}\right], \tag{3.14}$$

where σ_k^2 is the variance of the noise $\boldsymbol{\epsilon}_k$ injected at layer k. See Appendix A.1.5 for the proof.

To test our derived regularisers, in Figure 3.3 we show that models trained with R and GNIs have similar training profiles, whereby they have similar test-set loss and parameter Hessians throughout training, meaning that they have almost identical trajectories through the loss landscape. This implies that R is a good descriptor of the effect of GNIs and that we can use this term to understand the mechanism underpinning the regularising effect of GNIs. As we now show, it penalises neural networks that parameterize functions with higher frequencies in the Fourier domain; offering a novel lens under which to study GNIs.

3.4 Fourier Domain Regularisation

To link our derived regularisers to the Fourier domain, we use the connection between neural networks and Sobolev Spaces mentioned above. Recall that by Hornik (1991), we can only assume a sigmoid or piecewise linear neural network parameterises a function in

a weighted Sobolev space with measure μ, if we assume that the measure μ has compact support on a subset $\Omega \in \mathbb{R}^d$. As such, we equip our space with the *probability* measure $\mu(\mathbf{x})$, which we assume has compact support on some subset $\Omega \subset \mathbb{R}^d$ where $\mu(\Omega) = 1$. We define it such that $d\mu(\mathbf{x}) = p(\mathbf{x})d\mathbf{x}$ where $d\mathbf{x}$ is the Lebesgue measure and $p(\mathbf{x})$ is the data density function. Given this measure, we can connect the derivative of functions that are in the Hilbert-Sobolev space $W_\mu^{1,2}(\mathbb{R}^d)$ to the Fourier domain.

Theorem 3.3. *Consider a function, $f : \mathbb{R}^d \to \mathbb{R}$, with a d-dimensional input and a single output with $f \in W_\mu^{1,2}(\mathbb{R}^d)$ where μ is a probability measure which we assume has compact support on some subset $\Omega \subset \mathbb{R}^d$ such that $\mu(\Omega) = 1$. Let us assume the derivative of f, $D^\alpha f$, is in $L^2(\mathbb{R}^d)$ for some multi-index α where $|\alpha| = 1$. Then we can write that:*

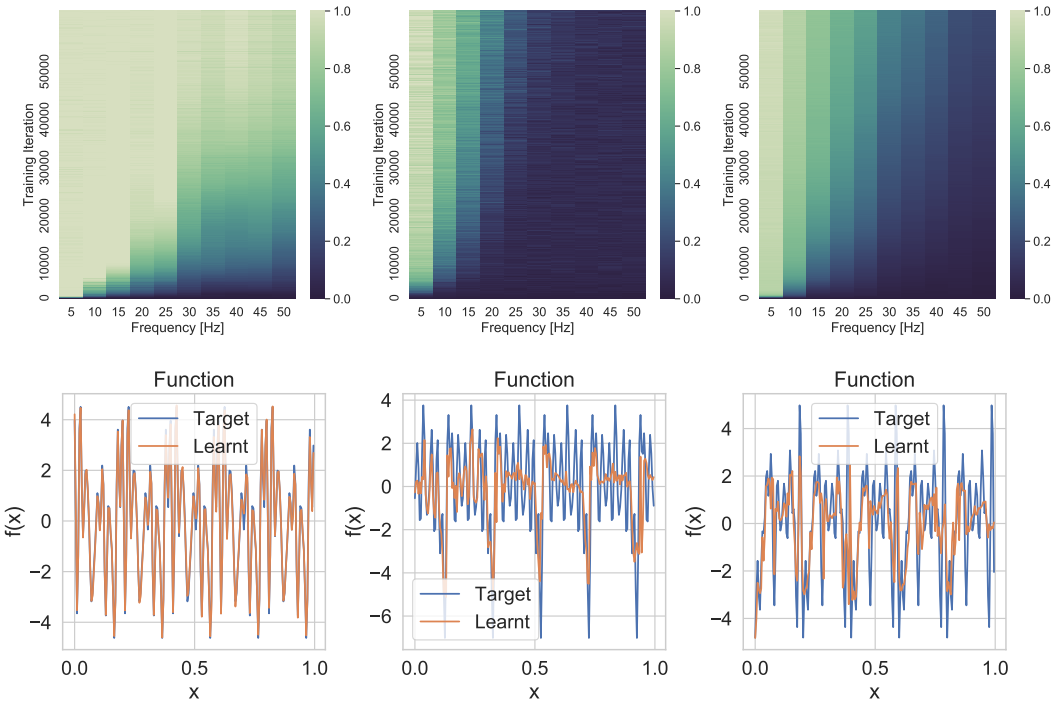
$$\sum_{|\alpha|=1} \|D^\alpha f\|^2_{L^2_\mu(\mathbb{R}^d)} = \int_{\mathbb{R}^d} \sum_{j=1}^{d} \left| \mathcal{G}(\boldsymbol{\omega}, j) \overline{\mathcal{G}(\boldsymbol{\omega}, j) * \mathcal{P}(\boldsymbol{\omega})} \right| d\boldsymbol{\omega} \tag{3.15}$$

$$\mathcal{G}(\boldsymbol{\omega}, j) = \boldsymbol{\omega}_j \mathcal{F}(\boldsymbol{\omega})$$

where $\mathcal{F}$ is the Fourier transform of f, $\mathcal{P}$ is the Fourier transform or the 'characteristic function' of the probability measure μ, j indexes over $\boldsymbol{\omega} = [\omega_1, \ldots, \omega_d]$, $*$ is the convolution operator, and $\overline{(\cdot)}$ is the complex conjugate.

See Appendix A.1.3 for the proof. Note that in the case where the dataset contains finitely many points, the integrals for the norms of the form $\|D^\alpha f_\theta\|^2_{L^2_\mu(\mathbb{R}^d)}$ are approximated by sampling a batch from the dataset which is distributed according to the presumed probability measure $\mu(\mathbf{x})$. Expectations over a batch thus approximate integration over $\mathbb{R}^d$ with the measure $\mu(\mathbf{x})$ and this approximation improves as the batch size grows. We can now use Theorem 3.3 to link R to the Fourier domain.

Regression Let us begin with the case of regression. Assuming differentiable and continuous activation functions, then the Jacobians within R are equivalent to the derivatives in Definition 3.1. Theorem 3.3 only holds for functions that have 1-D outputs, but we can decompose the Jacobians $\mathbf{J}_k$ as the derivatives of multiple 1-D output functions. Recall, that the i^{th} row of the matrix $\mathbf{J}_k$ is the set of partial derivatives of $f^\theta_{k,i}$, the function from layer k to the i^{th} network output, $i = 1...d_L$, with respect to the k^{th} layer activations.



(a) Baseline (b) Noise (c) R

Figure 3.4: As in Rahaman et al. (2019), we train 6-layer deep 256-unit wide ReLU networks trained to regress the function $\lambda(z) = \sum_i \sin(2\pi r_i z + \phi(i))$ with $r_i \in (5, 10, \ldots, 45, 50)$. We train these networks with no noise (Baseline), with GNIs of variance 0.1 injected into each layer except the final layer (Noise), and with the $R(\cdot)$ for regression in (3.11). The first row shows the Fourier spectrum (x-axis) of the networks as training progresses (y-axis) averaged over 10 training runs. Colours show each frequency's amplitude clipped between 0 and 1. The second row shows samples of randomly generated target functions and the function learnt by the networks.

Using this perspective, and the fact that each $f^{\theta}_{k,i} \in W^{1,2}_{\mu}(\mathbb{R}^{d_k})$ (d_k is the dimensionality of the k^{th} layer), if we assume that the probability measure of our space $\mu(\mathbf{x})$ has compact support, we use Theorem 3.3 to write:

$$
\begin{aligned}
R(\mathcal{B}; \mathbf{w}) &= \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim\mathcal{B}}\left[\sum_{k=0}^{L-1}\sigma_k^2 \sum_i \|\mathbf{J}_{k,i}(\mathbf{x})\|_F^2\right] \\
&= \frac{1}{2}\sum_{k=0}^{L-1}\sigma_k^2 \sum_i \mathbb{E}_{\mathbf{x}\sim\mathcal{B}}\left[\|\mathbf{J}_{k,i}(\mathbf{x})\|_F^2\right] \\
&\approx \frac{1}{2}\sum_{k=0}^{L-1}\sigma_k^2 \sum_i \sum_{|\alpha|=1} \|D^{\alpha} f^{\theta}_{k,i}\|^2_{L^2_{\mu}(\mathbb{R}^{d_k})} \\
&= \frac{1}{2}\sum_{k=0}^{L-1}\sigma_k^2 \sum_i \int_{\mathbb{R}^{d_k}} \sum_{j=1}^{d_k} \left|\mathcal{G}^{\theta}_{k,i}(\boldsymbol{\omega}, j)\overline{\mathcal{G}^{\theta}_{k,i}(\boldsymbol{\omega}, j) * \mathcal{P}(\boldsymbol{\omega})}\right| d\boldsymbol{\omega}
\end{aligned}
\tag{3.16}
$$

where $\mathbf{h}_0 = \mathbf{x}$, i indexes over output neurons, and $\mathcal{G}^\theta_{k,i}(\boldsymbol{\omega}, j) = \boldsymbol{\omega}_j \mathcal{F}^\theta_{k,i}(\boldsymbol{\omega})$, where $\mathcal{F}^\theta_{k,i}$ is the Fourier transform of the function $f^\theta_{k,i}$. The approximation comes from the fact that in SGD, as mentioned above, integration over the dataset is approximated by sampling mini-batches $\mathcal{B}$.

If we take the data density function to be the empirical data density, meaning that it is supported on the N points of the dataset $\mathcal{D}$ (i.e it is a set of δ-functions centered on each point), then as the size B of a batch $\mathcal{B}$ tends to N we can write that:

$$\lim_{B\to N} R(\mathcal{B}; \mathbf{w}) = \frac{1}{2}\sum_{k=0}^{L-1} \sigma_k^2 \sum_i \int_{\mathbb{R}^{d_k}} \sum_{j=1}^{d_k} \left| \mathcal{G}^\theta_{k,i}(\boldsymbol{\omega}, j) \overline{\mathcal{G}^\theta_{k,i}(\boldsymbol{\omega}, j) * \mathcal{P}(\boldsymbol{\omega})} \right| d\boldsymbol{\omega}. \tag{3.17}$$

Classification The classification setting requires a bit more work. Recall that our Jacobians are weighted by $\text{diag}(\mathbf{H}_L(\mathbf{x}))^\intercal$, which has positive entries that are less than 1 by Equation (3.13). We can define a new set of measures such that

$$d\mu_i(\mathbf{x}) = \text{diag}(\mathbf{H}_L(\mathbf{x}))_i^\intercal p(\mathbf{x}) d\mathbf{x}, \;\; i = 1 \dots d_L$$

. Because this new measure is positive, finite and still has compact support, Theorem 3.3 still holds for the spaces indexed by i: $W^{1,2}_{\mu_i}(\mathbb{R}^d)$.

Using these new measures, and the fact that each $f^\theta_{k,i} \in W^{1,2}_{\mu_i}(\mathbb{R}^{d_k})$, we can use Theorem 3.3 to write that for classification models:

$$\begin{aligned} R(\mathcal{B}; \mathbf{w}) &= \frac{1}{2}\sum_{k=0}^{L-1} \sigma_k^2 \sum_i \mathbb{E}_{\mathbf{x}\sim\mathcal{B}} \left[\text{diag}(\mathbf{H}_L(\mathbf{x}))_i^\intercal \|\mathbf{J}_{k,i}(\mathbf{x})\|_2^2 \right] \\ &\approx \frac{1}{2}\sum_{k=0}^{L-1} \sigma_k^2 \sum_i \sum_{|\alpha|=1} \|D^\alpha f^\theta_{k,i}\|^2_{L^2_{\mu_i}(\mathbb{R}^{d_k})} \\ &= \frac{1}{2}\sum_{k=0}^{L-1} \sigma_k^2 \sum_i \int_{\mathbb{R}^{d_k}} \sum_{j=1}^{d_k} \left| \mathcal{G}^\theta_{k,i}(\boldsymbol{\omega}, j) \overline{\mathcal{G}^\theta_{k,i}(\boldsymbol{\omega}, j) * \mathcal{P}_i(\boldsymbol{\omega})} \right| d\boldsymbol{\omega} \end{aligned} \tag{3.18}$$

Here $\mathcal{P}_i$ is the Fourier transform of the i^{th} measure μ_i and as before $\mathcal{G}^\theta_{k,i}(\boldsymbol{\omega}, j) = \boldsymbol{\omega}_j \mathcal{F}^\theta_{k,i}(\boldsymbol{\omega})$, where $\mathcal{F}^\theta_{k,i}$ is the Fourier transform of the function $f^\theta_{k,i}$. Again as the batch size increases to the size of the dataset, this approximation becomes exact.

For both regression and classification, GNIs, by way of R, induce a prior which favours

smooth functions with low-frequency components. This prior is enforced by the terms $\mathcal{G}^{\theta}_{k,i}(\boldsymbol{\omega}, j)$ which become large in magnitude when functions have high-frequency components, penalising neural networks that learn such functions. In Appendix A.2 we also show that this regularisation in the Fourier domain corresponds to a form of *Tikhonov regularisation*.

In Figure 3.4, we demonstrate empirically that networks trained with GNIs learn functions that don't overfit; with lower-frequency components relative to their non-noised counterparts.

A layer-wise regularisation Note that there is a recursive structure to the penalisation induced by R. Consider the layer-to-layer functions which map from a layer $k-1$ to k, $\mathbf{h}_k(\mathbf{h}_{k-1}(\mathbf{x}))$. With a slight abuse of notation, $\nabla_{\mathbf{h}_{k-1}}\mathbf{h}_k(\mathbf{x})$ is the Jacobian defined element-wise as:

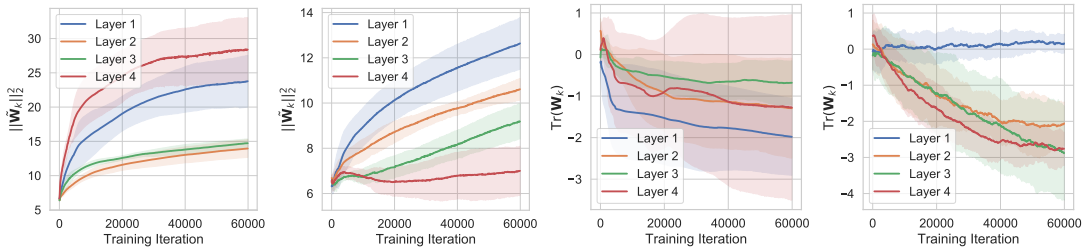
$$\left(\nabla_{\mathbf{h}_{k-1}}\mathbf{h}_k(\mathbf{x})\right)_{i,j} = \frac{\partial h_{k,i}}{\partial h_{k-1,j}(\mathbf{x})},$$

where as before $h_{k,i}$ is the i^{th} activation of layer k.

$\|\nabla_{\mathbf{h}_{k-1}}\mathbf{h}_k(\mathbf{x})\|_2^2$ is penalised k times in R as this derivative appears in $\mathbf{J}_0, \mathbf{J}_1 \dots \mathbf{J}_{k-1}$ due to the chain rule. As such, when training with GNIs, we can expect the norm of $\|\nabla_{\mathbf{h}_{k-1}}\mathbf{h}_k(\mathbf{x})\|_2^2$ to decrease as the layer index k increases (i.e the closer we are to the network output). By Theorem 3.3, and Equations (3.16), and (3.18), larger $\|\nabla_{\mathbf{h}_{k-1}}\mathbf{h}_k(\mathbf{x})\|_2^2$ correspond to functions with higher frequency components. Consequently, when training with GNIs the layer to layer function $\mathbf{h}_k(\mathbf{h}_{k-1}(\mathbf{x}))$ will have higher frequency components than the next layer's function $\mathbf{h}_{k+1}(\mathbf{h}_k(\mathbf{x}))$.

We measure this layer-wise regularisation in ReLU networks, using $\nabla_{\mathbf{h}_{k-1}}\mathbf{h}_k(\mathbf{x}) = \widetilde{\mathbf{W}}_k$. $\widetilde{\mathbf{W}}_k$ is obtained from the original weight matrix $\mathbf{W}_k$ by setting its i^{th} column to zero whenever the neuron i of the k^{th} layer is inactive. Also note that the inputs of ReLU network hidden layers, which are the outputs of another ReLU-layer, will be positive. Negative weights are likely to 'deactivate' a ReLU-neuron, inducing smaller $\|\widetilde{\mathbf{W}}_k\|_2^2$, and thus parameterising a lower frequency function. We use the trace of a weight matrix as an indicator for the 'number' of negative components.

In Figure 3.5 we demonstrate that $\|\widetilde{\mathbf{W}}_k\|_2^2$ *and* $\text{Tr}(\mathbf{W}_k)$ decrease as k increases for



(a) Baseline (b) GNI (c) Baseline (d) GNI

Figure 3.5: We use 6-layer deep 256-unit wide ReLU networks on the same dataset as in Figure 3.4 trained with (GNI) and without GNI (Baseline). In (a,b), for layers with square weight matrices, we plot $\|\widetilde{\mathbf{W}}_k\|_2^2$. In (c,d) we plot the trace of these layers' weight matrices $\text{Tr}(\mathbf{W}_k)$. For GNI models, as the layer index k increases, $\text{Tr}(\mathbf{W}_k)$ and $\|\widetilde{\mathbf{W}}_k\|_2^2$ decrease, indicating that each successive layer in these networks learns a function with lower frequency components than the past layer.

ReLU-networks trained with GNIs, indicating that each successive layer in these networks learns a function with lower frequency components than the past layer. This striation and ordering is clearly absent in the baselines trained without GNIs.

The Benefits of GNIs What does regularisation in the Fourier domain accomplish? The terms in R are the traces of the Gauss-Newton decompositions of the second order derivatives of the loss. By penalising this we are more likely to land in wider (smoother) minima (see Figure 3.3), which has been shown, contentiously (Dinh et al., 2017), to induce networks with better generalisation properties (Keskar et al., 2019; Jastrzębski et al., 2017). GNIs however, confer other benefits too.

Sensitivity. A model's weakness to input perturbations is termed the *sensitivity*. Rahaman et al. (2019) have shown empirically that classifiers biased towards lower frequencies in the Fourier domain are less sensitive, and there is ample evidence demonstrating that models trained with noised data are less sensitive (Liu et al., 2019; Li et al., 2018). The Fourier domain - sensitivity connection can be established by studying the *classification margins* of a model (see Appendix A.4).

Calibration. Given a network's prediction $\hat{y}(\mathbf{x})$ with confidence $\hat{p}(\mathbf{x})$ for a point $\mathbf{x}$, perfect calibration consists of being as likely to be correct as you are confident: $p(\hat{y} = y|\hat{p} = r) = r,\ \forall r \in [0, 1]$ (Dawid, 1982; DeGroot and Fienberg, 1983). In Appendix A.5 we show that models that are biased toward lower frequency spectra have lower 'capacity

measures', which measure model complexity and lower values of which have been shown empirically to induce better calibrated models (Guo et al., 2017). In Figure A.6.7 we show that this holds true for models trained with GNIs.

3.5 Related Work

Many variants of GNIs have been proposed to regularise neural networks. Poole et al. (2014) extend this process and apply noise to all computational steps in a neural network layer. Not only is noise applied to the layer input it is applied to the layer output and to the pre-activation function logits. The authors allude to explicit regularisation but only derive a result for a single layer auto-encoder with a single noise injection. Similarly, Bishop (1995) derive an analytic form for the explicit regulariser induced by noise injections on *data* and show that such injections are equivalent to Tikhonov regularisation in an unspecified function space.

Recently Wei et al. (2020) conducted similar analysis to ours, dividing the effects of Bernoulli dropout into *explicit* and *implicit* effects. Their work is built on that of Mele and Altarelli (1993), Helmbold and Long (2015), and Wager et al. (2013) who perform this analysis for linear neural networks. Arora et al. (2020) derive an explicit regulariser for Bernoulli dropout on the final layer of a neural network. Further, recent work by Dieng et al. (2018) shows that noise additions on recurrent network hidden states outperform Bernoulli dropout in terms of performance and bias.

3.6 Conclusion

In this work, we derived analytic forms for the explicit regularisation induced by Gaussian noise injections, demonstrating that the explicit regulariser penalises networks with high-frequency content in Fourier space. Further we show that this regularisation is not distributed evenly within a network, as it disproportionately penalises high-frequency content in layers closer to the network output. Finally we demonstrate that this regularisation in the Fourier domain has a number of beneficial effects. It induces training dynamics that preferentially land in wider minima, it reduces model sensitivity to noise, and induces bet-

ter calibration.

Acknowledgments

This research was directly funded by the Alan Turing Institute under Engineering and Physical Sciences Research Council (EPSRC) grant EP/N510129/1. AC was supported by an EPSRC Studentship. MW was supported by EPSRC grant EP/G03706X/1. UŞ was supported by the French National Research Agency (ANR) as a part of the FBIMATRIX (ANR-16-CE23-0014) project. SR gratefully acknowledges support from the UK Royal Academy of Engineering and the Oxford-Man Institute. CH was supported by the Medical Research Council, the Engineering and Physical Sciences Research Council, Health Data Research UK, and the Li Ka Shing Foundation

We thank Tomas Lazauskas, Jim Madge and Oscar Giles from the Alan Turing Institute's Research Engineering team for their help and support.

Impact Statement

This paper uncovers a new mechanism by which a widely used regularisation method operates and paves the way for designing new regularisation methods which take advantage of our findings. Regularisation methods produce models that are not only less likely to overfit, but also have better calibrated predictions that are more robust to distribution shifts. As such improving our understanding of such methods is critical as machine learning models become increasingly ubiquitous and embedded in decision making.

CHAPTER 4

ASYMMETRIC HEAVY TAILS AND IMPLICIT BIAS IN GAUSSIAN NOISE INJECTIONS

This work was written in collaboration with Xiaoyu Wang, Lingjiong Zhu, Mert Gürbüzbalaban, Chris Holmes, and Umut Şimşekli

4.1 Introduction . . . 44

4.2 Background . . . 46

4.2.1 Stable Distributions. . . . 46

4.2.2 Gaussian Noise Injections. . . . 49

4.3 The Implicit Effect of GNIs . . . 49

4.3.1 The Properties of the Accumulated Noise. . . . 50

4.3.2 Kurtosis of The Gradient Noise. . . . 52

4.3.3 Skewness of The Gradient Noise. . . . 54

4.4 An SDE Model for SGD with GNIs. . . . 54

4.4.1 Theoretical analysis of implicit bias. . . . 59

4.5 Further Experiments . . . 64

4.6 Conclusion . . . 65

Abstract

Gaussian noise injections (GNIs) are a family of simple and widely-used regularisation methods for training neural networks where one injects additive or multiplicative Gaussian noise to the network activations at every iteration of the optimisation algorithm, which is typically chosen as stochastic gradient descent (SGD). In this paper we focus on the so-called 'implicit effect' of GNIs, which is the effect of the injected noise on the dynamics of SGD. We show that this effect induces an *asymmetric heavy-tailed noise* on SGD gradient updates. In order to model this modified dynamics, we first develop a Langevin-like stochastic differential equation that is driven by a general family of *asymmetric* heavy-tailed noise. Using this model we then formally prove that GNIs induce an 'implicit bias', which varies depending on the heaviness of the tails and the level of asymmetry. Our empirical results confirm that different types of neural networks trained with GNIs are well-modelled by the proposed dynamics and that the implicit effect of these injections induces a bias that degrades the performance of networks.

4.1 Introduction

Noise injections are a family of methods that involve adding or multiplying samples from a noise distribution to the weights and activations of a neural network during training. The most commonly used distributions are Bernoulli distributions and Gaussian distributions (Srivastava et al., 2014; Poole et al., 2014) and the noise is most often inserted at the level of network activations.

Though the regularisation conferred by Gaussian noise injections (GNIs) can be observed empirically, and there have been many studies on the benefits of noising data (Bishop, 1995; Cohen et al., 2019; Webb, 1994), the mechanisms by which these injections operate are not fully understood. Recently, the *explicit* effect of GNIs, which is the added term to the loss function obtained when marginalising out the injected noise, has been characterised analytically (Camuto et al., 2020): it corresponds to a penalisation in the Fourier domain which improves model generalisation.

Here we extend this analysis and focus on the *implicit* effect of GNIs. This is the effect of the remaining noise that has been marginalised out when studying the explicit

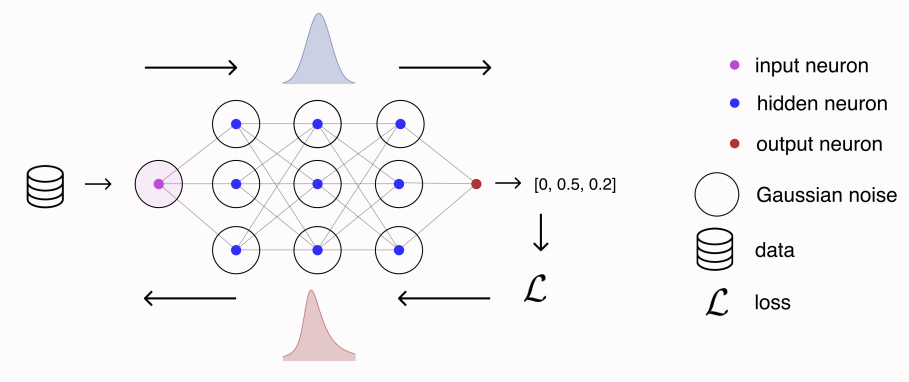


Figure 4.1: Illustration of the effect of GNIs *added* to a network's activations. Each colored dot represents a neuron's activations. We add GNIs, represented as circles, to each layer's activations bar the output layer. Perhaps counter-intuitively, though the forward pass experiences Gaussian noise, gradient updates in the backward pass experience heavy-tailed asymmetric noise.

effect. In particular we focus on the manner in which such noise alters the dynamics of Stochastic Gradient Descent (SGD) (Wei et al., 2020; Zhang et al., 2017). We show that the implicit effect is driven by an *asymmetric heavy-tailed noise* on the SGD gradient updates, as illustrated in Figure 4.1.

To study the effect of this gradient noise, we model the dynamics of SGD for a network experiencing GNIs by a stochastic differential equation (SDE) driven by an *asymmetric heavy-tailed* α-stable noise. We demonstrate that this model captures the dynamics of networks trained with GNIs and we show that the stationary distribution of this process becomes arbitrarily distant from the so-called *Gibbs measure*, whose modes exactly match the local minima of the loss function, as the gradient becomes increasingly heavy-tailed and asymmetric. Heavy-tailed and asymmetric gradient noise thus degrades network performance and this suggests that models trained with the full effect of GNIs will underperform networks trained solely with the explicit effect. We confirm this experimentally for a variety of dense and convolutional networks.[1]

4.2 Background

4.2.1 Stable Distributions.

The Generalised Central Limit Theorem (GCLT) (Gnedenko and Kolmogorov, 1954) states that for a sequence of independent and identically distributed (i.i.d.) random variables whose distribution has a power-law tail with index $0 < \alpha < 2$, the normalised sum converges to a heavy-tailed distribution called the α-stable distribution ($\mathcal{S}_\alpha$) as the number of summands grows. An α-stable distributed random variable X is denoted by $X \sim \mathcal{S}_\alpha(\sigma, \theta, \mu)$, where $\alpha \in (0, 2]$ is the *tail-index*, $\theta \in [-1, 1]$ is the *skewness* parameter, $\sigma \geq 0$ is the *scale* parameter, and $\mu \in \mathbb{R}$ is called the *location* parameter. The mean of X coincides with μ if $\alpha > 1$, and otherwise the mean of X is undefined. In this work, we always assume $\mu = 0$. The parameter θ is a measure of asymmetry. We say that X follows a *symmetric* α-stable distribution denoted as $\mathcal{S}\alpha\mathcal{S}(\sigma) = \mathcal{S}_\alpha(\sigma, 0, 0)$ if $\theta = 0$ (and $\mu = 0$). The parameter $\alpha \in (0, 2]$ determines the tail thickness of the distribution, and $\sigma > 0$ measures the spread of X around its mode. Note that when $\alpha < 2$, α-stable distributions have heavy tails such that their moments are finite only up to the order α.

The probability density function (p.d.f.) of an α-stable random variable, $\alpha \in (0, 2]$, does not have a closed-form expression except for a few special cases. When $\alpha = 1$ and $\alpha = 2$, the symmetric α-stable distribution reduces to the Cauchy and the Gaussian distributions, respectively, (cf. Section 1.1. in Samorodnitsky and Taqqu (1994)). By their flexibility, such distributions can model many complex stochastic phenomena for which exact analytic forms are intractable (Sarafrazi and Yazdi; Fiche et al., 2013).

Lévy Processes. A Lévy process (motion) is a stochastic process with independent, stationary increments. Formally, $\mathbf{L}_t$ is Lévy process if

- $\mathbf{L}_0 = 0$ almost surely;
- For any $t_0 < t_1 < \cdots < t_N$, the increments $\mathbf{L}_{t_n} - \mathbf{L}_{t_{n-1}}$ are independent, $n = 1, 2, \ldots, N$;
- The difference $\mathbf{L}_t - \mathbf{L}_s$ and $\mathbf{L}_{t-s}$ have the same distribution;

[1] See https://github.com/alexander-camuto/asym-heavy-tails-bias-GNI for all code.

- $\mathbf{L}_t$ is continuous in probability, i.e. for any $\delta > 0$ and $s \geq 0$, $\mathbb{P}(|\mathbf{L}_t - \mathbf{L}_s| > \delta) \to 0$ as $t \to s$.

The α-stable Lévy process is an important class of Lévy processes. In particular, for $\alpha \in (0, 2]$, let $\mathbf{L}_t^{\alpha,\theta}$ denote the d-dimensional α-stable Lévy process with independent components, i.e. each component is an independent scalar α-stable Levy motion (Duan, 2015) such that $\mathbf{L}_{t-s}^{\alpha,\theta}$ has the distribution $\mathcal{S}_\alpha((t-s)^{1/\alpha}, \theta, 0)$ for any $s < t$.

Stochastic Gradient Descent and Differential Equations. Let $\mathcal{D}$ be a training dataset composed of data-label pairs of the form $(\mathbf{x}, \mathbf{y})$, and let $\mathbf{w} \equiv \{\mathbf{W}_1, \ldots, \mathbf{W}_L\} \in \mathbb{R}^d$ be the d parameters of an L layer neural network in vector form. When a neural network operates on input data $\mathbf{x}$, we obtain the activations $\mathbf{h} \equiv \{\mathbf{h}_0, \ldots, \mathbf{h}_{L-1}\}$, where $\mathbf{h}_0 = \mathbf{x}$ and $\mathbf{h} \in \mathbb{R}^{n_0+\cdots+n_L}$ where n_i is the number of neurons in the i^{th} layer: we consider a non-linearity $\kappa : \mathbb{R} \to \mathbb{R}$, $\mathbf{h}_i(\mathbf{x}) = \kappa(\mathbf{W}_i \mathbf{h}_{i-1}(\mathbf{x}))$, where κ is applied element-wise to each coordinate of its argument. In supervised settings, our objective is to find the optimal parameters $\mathbf{w}_*$ that minimise the negative log-likelihood $-\log p_{\mathbf{w}}(\mathbf{y}|\mathbf{x})$ of the labels $\mathbf{y}$, given the parameters $\mathbf{w}$ and data $\mathbf{x}$:

$$\begin{aligned} \mathbf{w}_* &= \arg\min_{\mathbf{w}} \mathcal{L}(\mathcal{D}; \mathbf{w}), \\ \mathcal{L}(\mathcal{D}; \mathbf{w}) &:= -\mathbb{E}_{\mathbf{x},\mathbf{y}\sim\mathcal{D}} \left[\log p_{\mathbf{w}}(\mathbf{y}|\mathbf{x})\right] . \end{aligned} \tag{4.1}$$

SGD and its variants are the most prevalent optimisation routines that underpin the training of very large neural networks. Under SGD, we estimate equation (4.1) by sampling a *random* mini-batch of data-label pairs $\mathcal{B} \subset \mathcal{D}$,

$$\mathcal{L}(\mathcal{B}; \mathbf{w}) = -\mathbb{E}_{\mathbf{x},\mathbf{y}\sim\mathcal{B}} \left[\log p_{\mathbf{w}}(\mathbf{y}|\mathbf{x})\right] \approx \mathcal{L}(\mathcal{D}; \mathbf{w}). \tag{4.2}$$

SGD optimises this equation and approximates $\mathbf{w}_*$ using iterative parameter updates. At training step k

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta \nabla \mathcal{L}(\mathcal{B}_{k+1}; \mathbf{w}_k), \tag{4.3}$$

where η is the step-size for updates (the network's learning rate) (Robbins and Monro,

1951; Ruder, 2016).

Studying the dynamics of SGD allows us to understand the subtle effects that batching may have on neural network training. The similarities between the SGD algorithm and Langevin diffusions Roberts and Stramer (2002) have inspired many studies modelling the dynamics of SGD using stochastic differential equations (SDEs) under different noise conditions (Agapiou et al., 2014; Şimşekli et al., 2019a; Raginsky et al., 2017; Gao et al., 2018, 2020; Jastrzębski et al., 2017; Li et al., 2017). In this approach, one models the discrete SGD updates (4.3), as the discretisation of a continuous-time stochastic process, making assumptions about the properties of the 'noise' that drives this process (Mandt et al., 2016; Jastrzębski et al., 2017). This noise stems from the stochasticity in approximating the 'true' gradient over the dataset $\nabla\mathcal{L}(\mathcal{D};\mathbf{w}_k)$ with that of a mini-batch $\mathcal{B}$, $\nabla\mathcal{L}(\mathcal{B};\mathbf{w}_k)$. We denote this noise as,

$$U_{k+1}(\mathbf{w}) := \nabla\mathcal{L}(\mathcal{D};\mathbf{w}_k) - \nabla_{\mathbf{w}_k}\mathcal{L}(\mathcal{B}_{k+1};\mathbf{w}_k). \tag{4.4}$$

The most prevalent assumption is that the gradient noise admits a multi-variate Gaussian noise: $U_k(\mathbf{w}) \sim \mathcal{N}(0,\sigma^2\mathbf{I})$. This is rationalised by the central limit theorem, as the sum of estimation errors in equation (4.4) is approximately Gaussian for sufficiently large batches. Under this assumption, we can rewrite the SGD parameter update as:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta\nabla\mathcal{L}(\mathcal{D};\mathbf{w}_k) + \eta U_{k+1}(\mathbf{w}). \tag{4.5}$$

Then, one obtains the following continuous-time SDE to approximate the gradient updates (Welling and Teh, 2011; Mandt et al., 2016; Jastrzębski et al., 2017):

$$d\mathbf{w}_t = -\nabla\mathcal{L}(\mathcal{D};\mathbf{w}_t)dt + \sqrt{\eta\sigma^2}d\mathbf{B}_t, \tag{4.6}$$

where $\mathbf{B}_t$ is the Brownian motion in $\mathbb{R}^d$ and σ is the assumed noise variance for U. However, recent work suggests that the Gaussian assumption might not be always appropriate Gürbüzbalaban et al. (2021); Hodgkinson and Mahoney (2020), and connectedly, the gradient noise is observed to be heavy-tailed in different settings (Şimşekli et al., 2019; Zhou et al., 2020). Under this noise assumption, the corresponding SDE is such that $\mathbf{B}_t$ in (4.6)

is replaced with the *symmetric* stable process $\mathbf{L}_t^{\alpha,0}$ where $\theta = 0$:

$$d\mathbf{w}_t = -\nabla\mathcal{L}(\mathcal{D};\mathbf{w}_t)dt + \eta^{(\alpha-1)/\alpha}\sigma d\mathbf{L}_t^{\alpha,0}. \tag{4.7}$$

4.2.2 Gaussian Noise Injections.

GNIs are regularisation methods that consist of injecting Gaussian noise to the network activations. More precisely, let $\boldsymbol{\epsilon}$ be a collection of 'noise vectors' injected to the network activations at each layer *except the final layer*: $\boldsymbol{\epsilon} \equiv \{\boldsymbol{\epsilon}_0, \ldots, \boldsymbol{\epsilon}_{L-1}\}$, where $\boldsymbol{\epsilon}_i \in \mathbb{R}^{n_i}$, $\boldsymbol{\epsilon} \in \mathbb{R}^{n_0+\cdots+n_{L-1}}$ and n_i is the number of neurons in the i^{th} layer. We have two values for an activation: the soon-to-be noised value $\widehat{\mathbf{h}}_i$, and the subsequently noised value $\widetilde{\mathbf{h}}_i$. For a multi-layer perceptron (MLP),

$$\widehat{\mathbf{h}}_i(\mathbf{x}) = \kappa\left(\mathbf{W}_i\widetilde{\mathbf{h}}_{i-1}(\mathbf{x})\right), \qquad \widetilde{\mathbf{h}}_i = \widehat{\mathbf{h}}_i \circ \boldsymbol{\epsilon}_i\,, \tag{4.8}$$

where $\circ$ is some element-wise operation (e.g., addition or multiplication). For additive GNIs typically, $\boldsymbol{\epsilon}_i \sim \mathcal{N}\left(0, \sigma_i^2\mathbf{I}\right)$, and for multiplicative GNIs we often use $\boldsymbol{\epsilon}_i \sim \mathcal{N}\left(1, \sigma_i^2\mathbf{I}\right)$, where $\mathcal{N}$ is the Gaussian distribution. The 'accumulated' noise at each layer, induced by the noise injected to the layer *and* in previous layers is

$$\mathcal{E}_i(\mathbf{x};\mathbf{w},\boldsymbol{\epsilon}) = \widetilde{\mathbf{h}}_i(\mathbf{x}) - \mathbf{h}_i(\mathbf{x}). \tag{4.9}$$

Given that GNIs are commonly used as regularisation methods (Camuto et al., 2020; Dieng et al., 2018; Srivastava et al., 2014; Poole et al., 2014; Kingma et al., 2015; Bishop, 1995), our goal is to understand better the mechanisms by which they affect neural networks.

4.3 The Implicit Effect of GNIs

Recently, Camuto et al. (2020) showed that the effect of GNIs on the cost function can be expressed as a term $\Delta\mathcal{L}(\mathcal{B};\mathbf{w},\boldsymbol{\epsilon})$ that is added to the loss, i.e.,

$$\widetilde{\mathcal{L}}(\mathcal{B};\mathbf{w},\boldsymbol{\epsilon}) := \mathcal{L}(\mathcal{B};\mathbf{w}) + \Delta\mathcal{L}(\mathcal{B};\mathbf{w},\boldsymbol{\epsilon})\,, \tag{4.10}$$

where $\widetilde{\mathcal{L}}$ is the modified loss that SGD ultimately aims to minimise. The term $\Delta\mathcal{L}$ can be further broken down into explicit and implicit effects, as described in Section 6.1.

We build on the approach of Camuto et al. (2020) and define the explicit effect as the additional term obtained on the loss when we marginalise out the noise we have injected. It offers a consistently positive objective for gradient descent to optimise and we denote it as $\mathbb{E}_{\boldsymbol{\epsilon}}(\Delta\mathcal{L}(\mathcal{B};\mathbf{w},\boldsymbol{\epsilon}))$. The implicit effect is then the remainder of the terms marginalised out in the explicit effect:

$$E_{\mathcal{L}}(\mathcal{B};\mathbf{w},\boldsymbol{\epsilon}) := \Delta\mathcal{L}(\mathcal{B};\mathbf{w},\boldsymbol{\epsilon}) - \mathbb{E}_{\boldsymbol{\epsilon}}(\Delta\mathcal{L}(\mathcal{B};\mathbf{w},\boldsymbol{\epsilon})). \tag{4.11}$$

While the explicit effect focuses on the consistent and *non-stochastic* regularisation induced by GNIs, the implicit effect instead studies the effect of the *inherent stochasticity* of GNIs. This term does not offer a consistent objective for SGD to minimise. Rather we show that it affects neural network training by way of the *heavy-tailed and skewed* noise it induces on gradient updates.

Subsequently, we first characterise the tail properties of the noise accumulated during the forward pass. As this accumulated noise defines the implicit effect we can then use this result to show that gradients induced by the implicit effect are heavy-tailed and skewed and apt to be modelled by heavy-tailed and asymmetric α-stable noise.

4.3.1 The Properties of the Accumulated Noise.

Before considering the properties of the implicit effect gradients, we first need to study the noise that is accumulated during the *forward pass* of a neural network experiencing GNIs. Here we use asymptotic analysis to bound the moments of this noise, allowing us to characterise the tail properties of gradients using the broad class of *sub-Weibull* distributions that include a range of heavy-tailed distributions (Vladimirova et al., 2019, 2020; Kuchibhotla and Chakrabortty, 2018).

Definition 4.1. *(Asymptotic order) A positive sequence a_m is of the same order as another positive sequence b_m ($a_m \lesssim b_m$) if $\exists C > 0$ such that $\frac{a_m}{b_m} \leq C \ \forall m \in \mathbb{N}$. a_m is of the same order of magnitude as b_m ($a_m \asymp b_m$, i.e. 'asymptotically equivalent') if there exist some*

$c, C > 0$ such that: $c \leq \frac{a_m}{b_m} \leq C$ for any $m \in \mathbb{N}$.

Definition 4.2. *(Sub-Weibull distributions), : We say that a random variable X is sub-Weibull (Vladimirova et al., 2020; Kuchibhotla and Chakrabortty, 2018) with tail parameter r if $\|X\|_m \lesssim m^r, r > 0$, where $\|X\|_m := \mathbb{E}[|X|^m]^{\frac{1}{m}}$. In this case, we write $X \sim \text{subW}(r)$. Such distributions satisfy the tail bound $\mathbb{P}(|X| > x) \leq 2e^{-(\frac{x}{C})^{\frac{1}{r}}}$, where $C > 0$ is some constant. Note that for $r = \frac{1}{2}$ and $r = 1$ we recover the sub-Gaussian and sub-exponential families and that if $X \sim \text{subW}(r')$, then $X \sim \text{subW}(r)$ for $r > r'$.*

As r increases the tail distribution becomes heavier-tailed. To simplify our analysis we consider activation functions ϕ that obey the extended envelope property.

Definition 4.3 (Extended Envelope Property (Vladimirova et al., 2019):)**.** *A non-linear function $\kappa : \mathbb{R} \to \mathbb{R}$ is said to obey the extended envelope property if $\exists\ c_1, c_2 \geq 0,\ d_1, d_2 \geq 0$ such that:*

- *$|\kappa(x)| \geq c_1 + d_1|x|$, for any $x \in \mathbb{R}^+$ or $x \in \mathbb{R}^-$;*
- *$|\kappa(x)| \leq c_2 + d_2|x|$, for any $x \in \mathbb{R}$.*

Activation functions that obey this property, such as ReLU, are broadly moment preserving (Vladimirova et al., 2020). Using this property, we can characterise the moments for the l^{th} noised activation in a layer i, $\widetilde{h}_{i,l}(\mathbf{x})$.

Lemma 4.1. *For feed-forward neural networks with an activation function κ that obeys the extended envelope property, the noised activations at each layer $i < L - 1$, resulting from additive-GNIs $\boldsymbol{\epsilon}$ obey*

$$\left\|\widetilde{h}_{i,l}(\mathbf{x})\right\|_m \lesssim \sqrt{m}, \quad \text{for any } m \geq 1;\ l = 1, \ldots, n_i\,.$$

For multiplicative-GNIs we have,

$$\left\|\widetilde{h}_{i,l}(\mathbf{x})\right\|_m \lesssim m^{\frac{i+1}{2}}, \quad \text{for any } m \geq 1;\ l = 1, \ldots, n_i\,,$$

where n_i is the dimensionality of the i^{th} layer.

Lemma 4.1 shows that when the injected noise is additive, the noised activations at each layer will have *(sub)-Gaussian tails*. By equation (4.9), the accumulated noise $\mathcal{E}_i(\mathbf{x}; \mathbf{w}, \boldsymbol{\epsilon})$ will also have sub-Gaussian tails as the non-noised activations $\mathbf{h}(\mathbf{x})$ are deterministic and do not affect the asymptotic relationships of moments. See Figure B.1.1 of the Appendix for a demonstration that the activations experience Gaussian-like noise for additive-GNIs. For multiplicative-GNIs the noise at each layer, except the input layer which experiences Gaussian noise, behaves with a sub-Weibull tail. We demonstrate this behaviour in Figure B.1.2 of the Appendix.

We can now study the properties of the gradient noise induced by the implicit effect by taking the gradients of this forward pass noise.

4.3.2 Kurtosis of The Gradient Noise.

We characterise the gradient noise corresponding to $W_{i,l,j}$, the weight that maps from neuron l in layer $i-1$ to neuron j in layer i.

Theorem 4.1. *Consider a feed-forward neural network with an activation function κ that obeys the extended envelope property and a cross-entropy or mean-squared-error cost (see Appendix B.2). The gradient noise from additive-GNIs $\boldsymbol{\epsilon}$, has zero mean and has moments that obey for a pair $(\mathbf{x}, \mathbf{y})$:*

$$\left\| \frac{\partial E_{\mathcal{L}}((\mathbf{x}, \mathbf{y}); \mathbf{w}, \boldsymbol{\epsilon})}{\partial W_{i,l,j}} \right\|_m \lesssim m, \quad \text{for any } m \geq 1,$$

where $E_{\mathcal{L}}((\mathbf{x}, \mathbf{y}); \mathbf{w}, \boldsymbol{\epsilon})$ is defined in (4.11). For multiplicative GNIs, we have

$$\left\| \frac{\partial E_{\mathcal{L}}((\mathbf{x}, \mathbf{y}); \mathbf{w}, \boldsymbol{\epsilon})}{\partial W_{i,l,j}} \right\|_m \lesssim m^{\frac{L+i}{2}}, \quad \text{for any } m \geq 1,$$
$$i = 1, \ldots, L;\ l = 1, \ldots, n_{i-1};\ j = 1, \ldots, n_i.$$

For the additive noise, these bounds infer that the gradient noise at each layer will have *sub-exponential tails*. For the multiplicative case, gradient noise will be *sub-Weibull*, with a tail parameter that increases with i the layer index. Unlike the forward pass, which experienced noise bounded in its tails by a Gaussian, the backward pass experiences noise

that is bounded in its tails by heavy-tailed Weibull distributions with tail parameter $r \geq 1$.

Remark 4.2. *The bounds defined by Lemma 4.1 and Theorem 4.1 become an asymptotic equivalence ($\asymp$) in the case of 1-D data and 1-neuron-wide neural networks, i.e. the bounds are maximally tight in this case.*

Our result applies to the gradient for a single pair $(\mathbf{x}, \mathbf{y})$. During SGD, we take the mean gradient across a batch $\mathcal{B}$ of size B. To study the tail distribution of this mean gradient, we restate in a simplified manner Kuchibhotla and Chakrabortty (2018)'s generalisation of the Bernstein inequality for zero-mean sub-Weibull random variables in Theorem 4.3.

Theorem 4.3 (Theorem 3.1 in Kuchibhotla and Chakrabortty (2018))**.** *Let $X_1, \ldots, X_B$ be independent mean-zero sub-Weibull random variables with a shared tail parameter $p \geq 1$. Then, for every $x \geq 0$, we have*

$$\mathbb{P}\left\{\left|(1/B)\sum_{i=1}^{B} X_i\right| \geq x\right\} \leq 2\exp\left[-\min\left(\frac{Bx^2}{C\sum_{i=1}^{B}\|X_i\|_{\psi_2}^2}, \frac{Bx^{\frac{1}{\theta}}}{L\max_i\|X_i\|_{\psi_{\frac{1}{p}}}}\right)\right],$$

where $C, L > 0$ are constants that depend on the tail parameter and where

$$\|X\|_{\psi_{\frac{1}{p}}} = \inf\left\{\nu \geq 0 : \mathbb{E}\left[(|X|/\nu)^{\frac{1}{p}} \leq 1\right]\right\} \tag{4.12}$$

is the 'sub-Weibull norm'.

This theorem states that the the tails of the mean of zero-mean i.i.d. sub-Weibull random variables are produced by a single variable, say X_i, with the maximal sub-Weibull norm $\|X_i\|_{\psi_{\frac{1}{p}}} = \inf\{\nu \geq 0 : \mathbb{E}[(|X_i|/\nu)^{\frac{1}{p}} \leq 1]\}$, i.e., the one with the heaviest tails. Assuming gradients are independent across data points, we can use this inequality to bound the tail probability for the sum of our zero-mean gradients. If a single gradient is sufficiently heavy-tailed, then the mean across the batch will also be heavy-tailed.

In Figure 4.2 we show that the implicit effect gradient, averaged over *the entire dataset* $\mathcal{D}$ (i.e., the largest batch-size possible), is heavy-tailed, unlike the forward pass. To calculate these gradients, we estimate the explicit regulariser in equation (4.11) using Monte Carlo sampling, $\mathbb{E}_{\boldsymbol{\epsilon}}(\Delta\mathcal{L}(\mathcal{D}; \mathbf{w}, \boldsymbol{\epsilon})) \approx \frac{1}{M}\sum_{m=0}^{M}\Delta\mathcal{L}(\mathcal{D}; \mathbf{w}, \boldsymbol{\epsilon}_m)$, similarly to Wei et al. (2020);

Camuto et al. (2020). We show these results for multiplicative-GNIs in Figure B.1.3 of the Appendix. In this setting as well, the backward pass experiences *heavy-tailed* noise from GNIs.

4.3.3 Skewness of The Gradient Noise.

In the proof of Theorem 4.1 we decompose $\partial E_{\mathcal{L}}(\cdot)/\partial W_{i,l,j}$ as $(\partial E_{\mathcal{L}}(\cdot)/\partial \widetilde{h}_{i,j})\cdot(\partial \widetilde{h}_{i,j}/\partial W_{i,l,j})$, where $\widetilde{h}_{i,j}$ is the (noised) activation of the j^{th} neuron in the i^{th} layer. Both these derivatives are likely to be skewed due to the asymmetry of the activation functions and their gradients.

Ignoring potential correlations between variables, we observe that product of two zero-mean skewed independent variables X and Y is also skewed

$$|\text{skew}(XY)| = \left|\mathbb{E}\left[X^3\right]\mathbb{E}\left[Y^3\right]\right| > 0.$$

The skewness of the derivatives in the backward pass will induce *skewed* gradient noise, as seen in Figure 4.2. Though correlations between gradients could also cause skewness, we show that this is not the case in Appendix B.3.

4.4 An SDE Model for SGD with GNIs.

In this section, we will analyse the effects of the skewed and heavy-tailed noise on the SGD dynamics. Recall that the modified loss function by the GNIs is the sum of the explicit regulariser and the original loss over the dataset $\mathcal{D}$,

$$\mathbb{E}_{\boldsymbol{\epsilon}}\left(\widetilde{\mathcal{L}}(\mathcal{D};\mathbf{w},\boldsymbol{\epsilon})\right) = \mathcal{L}(\mathcal{D};\mathbf{w}) + \mathbb{E}_{\boldsymbol{\epsilon}}\left(\Delta\mathcal{L}(\mathcal{D};\mathbf{w},\boldsymbol{\epsilon})\right),$$

and the SGD recursion takes the following form:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta\nabla\,\mathbb{E}_{\boldsymbol{\epsilon}}\left(\widetilde{\mathcal{L}}(\mathcal{D};\mathbf{w}_k,\boldsymbol{\epsilon})\right) + \eta U_k(\mathbf{w}), \tag{4.13}$$

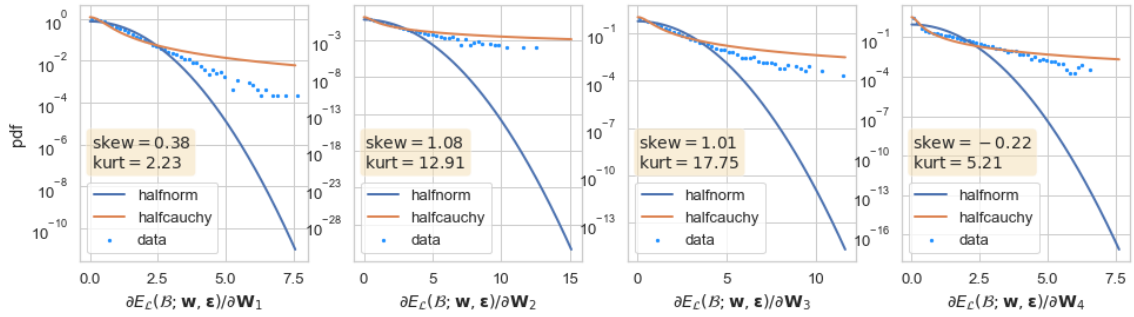


Figure 4.2: We measure the skewness and kurtosis *at initialisation* of the gradients noise accrued on networks weights during the backward pass for additive-GNIs. The model is a 4-layer-256-unit-wide MLP trained to regress $\lambda(x) = \sum_i \sin(2\pi q_i x + \phi(i))$ with $q_i \in (5, 10, \ldots, 45, 50), x \in \mathbb{R}$. We plot the probability density function (p.d.f.) of positive samples, comparing against half-normal and half-Cauchy distributions. Each blue point represents the mean gradient noise over the entire dataset $\mathcal{D}$ of an individual weight in a layer i. This gradient noise is *skewed and heavy-tailed*, with a p.d.f. that is more Cauchy-like than Gaussian.

where $U_{k+1}(\mathbf{w})$ is given as follows:

$$\nabla \left[\mathbb{E}_{\boldsymbol{\epsilon}} \left(\widetilde{\mathcal{L}}(\mathcal{D}; \mathbf{w}_k, \boldsymbol{\epsilon})\right)\right] \quad - \nabla \left[\mathbb{E}_{\boldsymbol{\epsilon}} \left(\widetilde{\mathcal{L}}(\mathcal{B}_{k+1}; \mathbf{w}_k, \boldsymbol{\epsilon})\right) + E_{\mathcal{L}}(\mathcal{B}_{k+1}; \mathbf{w}_k, \boldsymbol{\epsilon})\right]. \tag{4.14}$$

To ease the notation, let us denote the modified loss function by $f(\mathbf{w}) := \mathbb{E}_{\boldsymbol{\epsilon}} \left(\widetilde{\mathcal{L}}(\mathcal{D}; \mathbf{w}, \boldsymbol{\epsilon})\right)$.

Remark 4.4. *Note that when the gradients are computed over the whole dataset $\mathcal{D}$, the gradient noise solely stems from the implicit effect, $U_k(\mathbf{w}) = -\nabla [E_{\mathcal{L}}(\mathcal{D}; \mathbf{w}_k, \boldsymbol{\epsilon})]$.*

Recent studies have proven that heavy-tailed behaviour can already emerge in stochastic optimisation (without GNIs) (Hodgkinson and Mahoney, 2020; Gürbüzbalaban et al., 2021; Gürbüzbalaban and Hu, 2021), which can further result in an overall heavy-tailed behaviour in the gradient noise, as empirically reported in Şimşekli et al. (2019a); Zhang et al. (2020a); Zhou et al. (2020). Accordingly, Şimşekli et al. (2019a); Zhou et al. (2020) proposed modelling the gradient noise by using a centred *symmetric* α-stable noise, which then paved the way for modelling the SGD dynamics by using an SDE driven by a symmetric α-stable process (see (4.7)).

We take a similar route for modelling the trajectories of SGD with GNIs; however, due to the skewness arising from the GNIs, the symmetric noise assumption is not appropriate for our purposes. Hence, we propose modelling the skewed gradient noise (4.14) by using an *asymmetric* α-stable noise, which aims at modelling both the heavy-tailed behaviour

and the asymmetries at the same time. In particular, we consider an SDE driven by an asymmetric stable process and its Euler discretisation as follows:

$$d\mathbf{w}_t = -\nabla f(\mathbf{w}_t)dt + \varepsilon d\mathbf{L}_t^{\alpha,\theta}, \tag{4.15}$$

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta_{k+1}\nabla f(\mathbf{w}_k) + \varepsilon\eta_{k+1}^{1/\alpha}\Delta\mathbf{L}_{k+1}^{\alpha,\theta}, \tag{4.16}$$

where $\theta = (\theta_i, 1 \leq i \leq d)$ is the d-dimensional skewness and each coordinate can have its idiosyncratic skewness θ_i. $\mathbf{L}_t^{\alpha,\theta} = (L_t^{\alpha,\theta_1}, \ldots, L_t^{\alpha,\theta_d})$ is a d-dimensional asymmetric α-stable Lévy process with independent components, and ε encapsulates all the scaling parameters. Furthermore, $(\eta_k)_k$ denotes the sequence of step-sizes, which can be taken as constant or decreasing, and finally $(\Delta\mathbf{L}_k^{\alpha,\theta})_k$ is a sequence of i.i.d. random vectors where each component of $\Delta\mathbf{L}_k^{\alpha,\theta}$ is i.i.d. with $\mathcal{S}_\alpha(1, \theta_i, 0)$. We then propose the discretised process (4.16) as a proxy to the original recursion (4.13) and we will directly analyse the theoretical properties of (4.16). Note that our approach strictly extends Şimşekli et al. (2019a), which appears as a special case when $\theta = 0$.

Before deriving our theoretical results, we first verify empirically that the proxy dynamics (4.16) are indeed a good model for representing (4.13). We ascertain that $\Delta\mathbf{L}_k^{\alpha,\theta}$ is sufficiently general in the sense that it can capture the gradient noise induced by the implicit effect even when there is no batching noise (when the batch size approaches the size of the dataset for example, see Remark 4.4). As a first line of evidence, in Figure B.1.4 of the Appendix we model $\nabla E_{\mathcal{L}}(\cdot)$ as being drawn from a univariate $\mathcal{S}_\alpha$. The equivalent $\mathcal{S}_\alpha$ distributions are skewed ($|\theta| > 0$) and heavy-tailed ($\alpha < 2$), demonstrating that $\mathbf{L}_t^{\alpha,\theta}$ captures the core properties of the implicit effect gradients highlighted in Section 4.3. To illustrate this more clearly, in Figure 4.3 we use symmetric sub-Weibull distributions with $r = 0.8, 1, 2$, and fit $\mathcal{S}_\alpha$ and Gaussians ($\mathcal{N}$) using maximum likelihood (MLE) from 10^4 samples. We plot the MLE densities, and clearly MLE $\mathcal{S}_\alpha$ ($\alpha < 2$) better model the tails of the sub-Weibulls than a Gaussian distribution, even for $r = 0.8$ which is close to Gaussian tails ($r = 0.5$). The $\mathcal{S}_\alpha$ modelling of the tails improves as r increases. This further illustrates the appropriateness of our noise model.

In Figure 4.4, we further show that the gradient noise of the implicit effect and gradient noise drawn from an equivalent $\mathcal{S}_\alpha$ distribution will have similar effects on gradient descent.

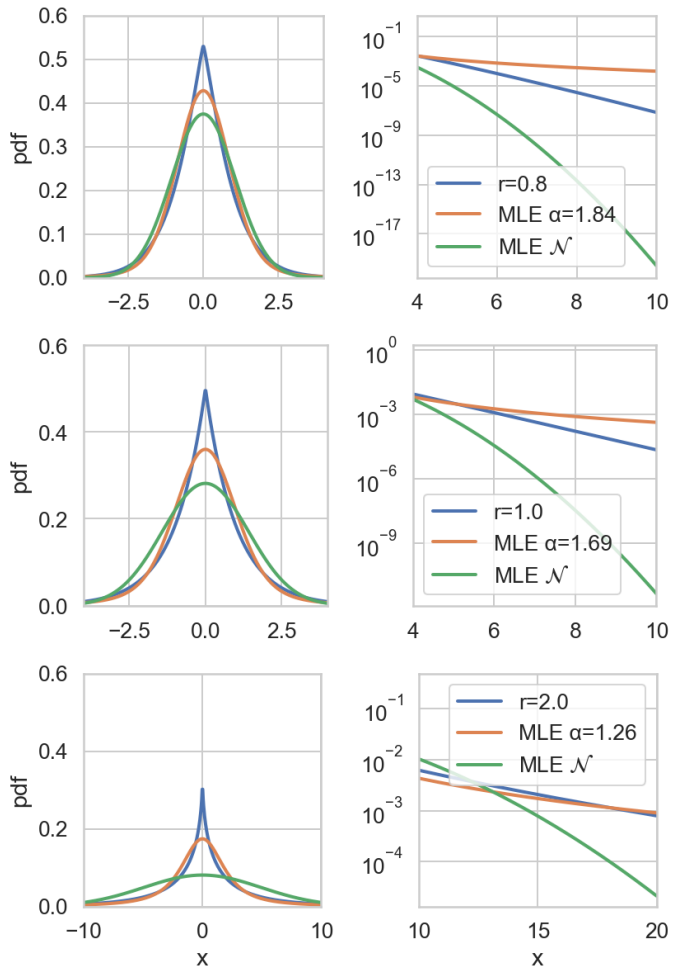


Figure 4.3: [left col.] sub-Weibull pdfs (r=0.8, 1.0, 2.0), and pdfs of MLE fitted $\mathcal{S}_\alpha$ and $\mathcal{N}$. [right col.] pdf in the tails ($x > 4$ and $x > 10$, note log y-axis).

We sample M GNI samples and evaluate:

$$\widetilde{\mathcal{L}}_M(\mathcal{D}; \mathbf{w}, \boldsymbol{\epsilon}) = (1/M) \sum_{m=0}^{M} \widetilde{\mathcal{L}}(\mathcal{D}; \mathbf{w}, \boldsymbol{\epsilon}_m). \tag{4.17}$$

The objective is over the entire dataset such that we eliminate noise from the batching process. M allows us to control the 'degree' to which the implicit effect is marginalised out. $M = 1$ corresponds to the usual training with GNI and larger values of M mimic the effects of marginalising out the implicit effect. We model $\nabla E_{\mathcal{L}}$ for $M = 1$ as being drawn from a univariate $\mathcal{S}_\alpha$ or a univariate normal distribution and estimate distribution parameters using maximum likelihood estimation, as in Nolan (2001), at each training iteration. We add draws from the estimated distributions to the gradients of $M = 16$ models to mimic the combined implicit and explicit effects. $M = 16$ models with the added $\mathcal{S}_\alpha$ noise have

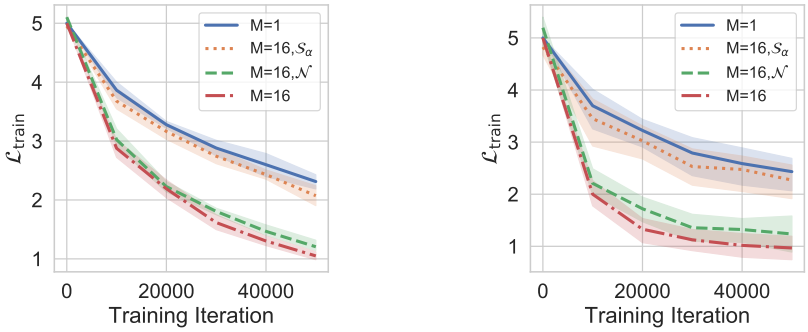


Figure 4.4: We train the networks in Figure 4.2 on $\widetilde{\mathcal{L}}_M$ (4.17) for the sinusoidal toy-data with additive GNIs [left] and multiplicative GNIs [right]. We fit univariate $\mathcal{S}_\alpha$ and univariate $\mathcal{N}$ via maximum likelihood (Nolan, 2001) to $\nabla E_{\mathcal{L}}(\cdot)$ for $M = 1$ models *at each training step*. We add draws from these distributions to the gradients of $M = 16$ models and plot the training loss ($\mathcal{L}_{\text{train}}$). Shading is the standard deviation over 5 random seeds.

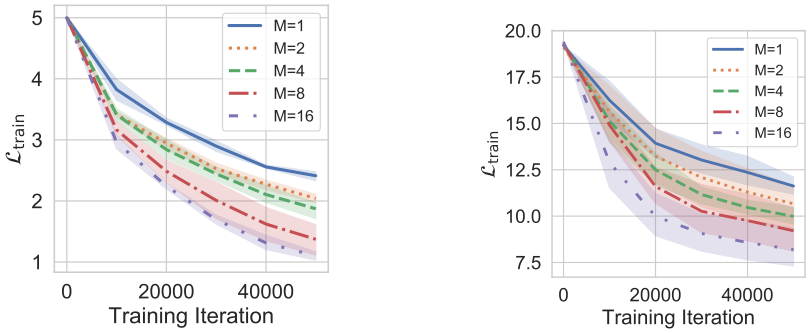


Figure 4.5: We train the networks in Figure 4.2 on $\widetilde{\mathcal{L}}_M$ (4.17) for the sinusoidal toy-data with additive GNIs [left] and MNIST with multiplicative GNIs [right]. We plot the training loss ($\mathcal{L}_{\text{train}}$). Shading is the standard deviation over 5 random seeds.

the same training path as $M = 1$ models, whereas those with Gaussian gradient noise do not. Thus, $\mathcal{S}_\alpha$ distributions are able to faithfully capture the dynamics induced by the implicit effect on gradient descent.

In these same experiments $M = 16$ models outperform $M = 1$ models on training data. We refine this study for a greater range of M values in Figure 4.5. As M increases, performance of models on training data improves gradually, suggesting that the implicit effect degrades performance. Further, in Figure 4.4, $M = 16$ models *trained with Gaussian noise* added to gradients outperform $M = 1$ models, suggesting that the heavy-tails and skew of the implicit effect gradients are responsible for this performance degradation. We now study this apparent bias.

4.4.1 Theoretical analysis of implicit bias.

Due to their heavy-tailed nature, stable processes have significantly different statistical properties from those of their Brownian counterparts. Their trajectories have a countable number of discontinuities, called *jumps*, whereas Brownian motion is continuous almost everywhere. With these jumps the process can escape from 'narrow' basins and spend more time in 'wider' basins (see Appendix B.6 for the definition of width). Theoretical results demonstrating this have been provided for symmetric stable processes ($\theta = 0$) (Şimşekli et al., 2019a). By translating the related metastability results from statistical physics (Imkeller and Pavlyukevich, 2008) to our context, in Appendix B.6 we illustrate that this also holds for SDEs driven by asymmetric stable processes ($\theta \neq 0$). In this sense, the SDE (4.15) is 'biased' towards wider basins.

While driving SGD iterates towards wider minima could be beneficial, we now show that heavy tails can also introduce an undesirable bias and that this bias is magnified by asymmetries. To quantify this bias, we focus on the invariant measure (i.e., the stationary distribution) of the Markov process (4.16) and investigate its modes (i.e., its local maxima), around where the process resides most of the time.

In a statistical physics context, Dybiec et al. (2007) empirically illustrated that the asymmetric stable noise can cause 'shifts', in the sense that the modes of the stationary distribution of (4.16) can shift away from the true local minima of f, which are of our interest as our aim is to minimise f. They further illustrated that such shifts can be surprisingly large when α gets smaller and $|\theta|$ gets larger. We illustrate this outcome by reproducing one of the experiments provided in Dybiec et al. (2007) in Figure 4.6. Here, we consider a one-dimensional problem with the quartic potential $f(w) = w^4/4 - w^2/2$, and simulate (4.16) for 10K iterations with constant step-size $\eta_k = 0.001$ and $\varepsilon = 1$. By using the generated iterates, we estimate the density of the invariant measure of (4.16) by using the kernel density estimator provided in scikit-learn (Pedregosa et al., 2011), for different values of α and θ. When α is larger (left), the heavy-tails cause a shift in the modes of the invariant measure, where these shifts become slightly larger with increasing asymmetries ($|\theta| > 0$). When the tails are heavier (right), we observe a much stronger interaction between α and θ, and observe *drastic* shifts as the asymmetry is increased.

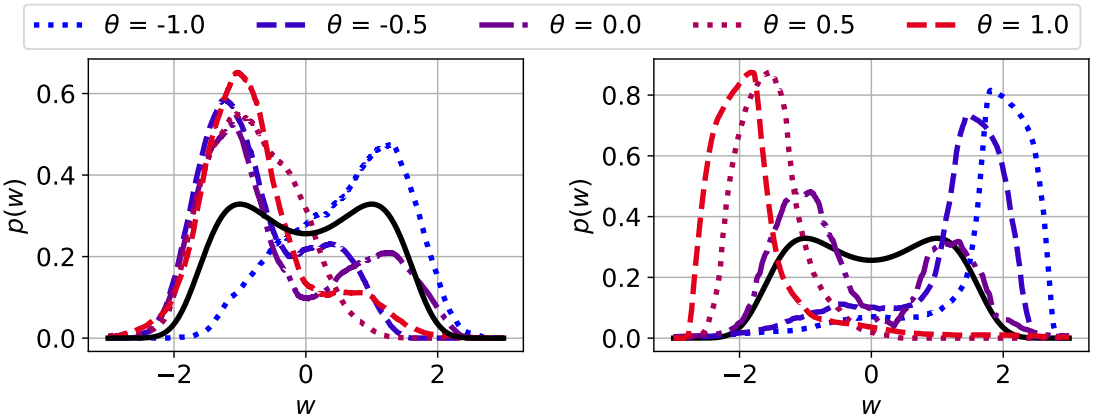


Figure 4.6: The stationary distributions of (4.16) with $\alpha = 1.9$ (left) and $\alpha = 1.1$ (right). The solid black line represents the density of the Gibbs measure $\exp(-f(w))$ with $f(w) = w^4/4 - w^2/2$.

From an optimisation perspective, these results are rather unsettling as they imply that SGD might spend most of its time in regions that are arbitrarily far from the local minima of the objective function f, since the high probability regions of its stationary distribution might be shifted away from the local minima of interest. In the symmetric case ($\theta = 0$), this observation has been formally proven in Sliusarenko et al. (2013) when f is chosen as the one dimensional quartic potential of Figure 4.6 and when $\alpha = 1$. A direct quantification of such shifts is non-trivial; and in the presence of asymmetries ($\theta \neq 0$), even further difficulties emerge.

In a recent study Şimşekli et al. (2020) focused on eliminating the undesired bias introduced by *symmetric* stable noise in SGD with momentum (Qian, 1999), and proposed an indirect way to ensure that modes of the stationary distribution exactly match the objective function's local minima. They developed a 'modified' SDE whose invariant distribution can be proven to be the *Gibbs measure*, denoted by $\pi(d\mathbf{w})$, which is a probability measure that has a density proportional to $\exp(-Cf(\mathbf{w}))$ for some $C > 0$. Clearly, all the local maxima of this density coincide with the local minima of the function f; hence, their approach eliminates the possibility of a shift in the modes by imposing a stronger condition which controls the entire invariant distribution.

By following a similar approach, we will bound the gap between the invariant measure of (4.16) and the Gibbs measure in terms of the tail index α and the skewness θ, using it as a quantification of the bias induced by the asymmetric heavy-tailed noise. In particular, for any sufficiently regular test function g, we consider its expectation under

the Gibbs measure $\nu(g) := \int g(\mathbf{w})\pi(d\mathbf{w})$, and its sample average computed over (4.16), i.e., $\nu_N(g) := \frac{1}{H_N}\sum_{k=1}^{N}\eta_k g(\mathbf{w}_k)$, where $H_N = \sum_{k=1}^{N}\eta_k$. We then bound the *weak error*: $|\nu(g) - \lim_{N\to\infty}\nu_N(g)|$, whose convergence to zero is sufficient for ensuring the modes do not shift.

We derive this bound for our case in three steps: (i) We first link the discrete-time process (4.16) to its continuous-time limit (4.15) by directly using the results of Panloup (2008). (ii) We then design a modified SDE that has the unique invariant measure as the Gibbs measure, with all the modes matching those of the loss function (Theorem 4.5). (iii) Finally, we show that the SDE (4.15) is a poor numerical approximation to the modified SDE, and we develop an upper-bound for the approximation error (Theorem 4.6).

To address (ii), we introduce a modification to (4.15), coined *asymmetric fractional Langevin dynamics*:

$$d\mathbf{w}_t = b(\mathbf{w}_{t-}, \alpha, \theta)dt + \varepsilon d\mathbf{L}_t^{\alpha,\theta}\,. \tag{4.18}$$

where, the drift function $b(\mathbf{w}, \alpha, \theta) := ((b(\mathbf{w}, \alpha, \theta))_i, 1 \le i \le d)$ is defined as follows:

$$(b(\mathbf{w}, \alpha, \theta))_i = \frac{\varepsilon^\alpha}{\varphi(\mathbf{w})}\mathcal{D}_{w_i}^{\alpha-2,-\theta_i}(\partial_{w_i}\varphi(\mathbf{w})), \tag{4.19}$$

where $\theta_i \in (-1, 1)$, $1 \le i \le d$, $1 < \alpha < 2$, and $\varphi(\mathbf{w}) := e^{-\varepsilon^{-\alpha}f(\mathbf{w})}$. Here, the operator $\mathcal{D}^{\alpha-2,-\theta_i}$ denotes a *Riesz-Feller type fractional derivative* (Gorenflo and Mainardi, 1998; Mainardi et al., 2001) whose exact (and rather complicated) definition is not essential in our problematic, and is given in Appendix B.5 in order to avoid obscuring the main results. The next theorem states that the SDE (4.18) targets the Gibbs measure.

Theorem 4.5. *The Gibbs measure* $\pi(d\mathbf{w}) \propto \exp(-\varepsilon^{-\alpha}f(\mathbf{w}))d\mathbf{w}$ *is an invariant distribution of* (4.18). *If* $b(\mathbf{w}, \alpha, \theta)$ *is Lipschitz continuous in* $\mathbf{w}$, *then* $\pi(d\mathbf{w})$ *is the unique invariant distribution of* (4.18).

This theorem states that the use of the modified drift b in place of $-\nabla f$, prevents any potential shifts in the modes of the invariant measure.

The fractional derivative in (4.19) is a non-local operator that requires the knowledge

of the full function, and does not admit a closed-form expression. In the next step, we develop an approximation scheme for the drift b in (4.18), and show that the gradient $-\nabla f$ in (4.16) appears as a special case of this scheme. To simplify notation, we consider the one-dimensional case ($d = 1$); however, our results can be easily extended to multivariate settings by applying the same approach to each coordinate. Hence, in the general case the bounds will scale linearly with d. We define the following approximation for b:

$$b_{h,K}(w, \alpha, \theta) := \frac{\varepsilon^\alpha}{\varphi(w)} \Delta_{h,K}^{\alpha-2,-\theta}(\partial_w \varphi(w)), \tag{4.20}$$

where, for an arbitrary function ψ, we have

$$\Delta_{h,K}^{\gamma,-\theta}\psi(w) := \frac{c_\gamma}{h^\gamma} \sum_{k=-K}^{K} (1 + \theta\,\mathrm{sign}(k))\tilde{g}_{\gamma,k}\psi(w - kh).$$

Here, sign denotes the sign function, $h > 0$, $K \in \mathbb{N} \cup \{0\}$, $c_\gamma := 1/(2\cos(\gamma\pi/2))$, and $\tilde{g}_{\gamma,k} := (-1)^k \Gamma(-\gamma+k)/\Gamma(k+1)\Gamma(-\gamma)$. This approximation is designed in the way that we recover the original drift b as $h \to 0$ and $K \to \infty$ for sufficiently regular φ. It is clear that when we set $K = 0$ and $h = h_0 := [2\varepsilon^{-\alpha}\cos((\alpha-2)\pi/2)]^{1/(2-\alpha)}$, we have $b_{h,K}(w, \alpha, \theta) = -\partial_w f(w)$. In the multidimensional case, where we apply this approximation to each coordinate, the same choice of K and h gives us the original gradient $-\nabla f$; hence, we fall back to the original recursion (4.16). By considering the recursion with this approximate drift

$$\tilde{\mathbf{w}}_{n+1} = \tilde{\mathbf{w}}_n + \eta_{n+1} b_{h,K}(\tilde{\mathbf{w}}_n, \alpha, \theta) + \varepsilon \eta_{n+1}^{1/\alpha} \Delta \mathbf{L}_{n+1}^{\alpha,\theta},$$

and the corresponding sample averages $\tilde{\nu}_N(g) := \frac{1}{H_N}\sum_{k=1}^N \eta_k g(\tilde{\mathbf{w}}_k)$[2], we are ready to state our error bound. We believe this result is interesting on its own, and would be of further interest in statistical physics and applied probability. To avoid obscuring the result, we state the required assumptions in the Appendix, which mainly require decreasing step-size and ergodicity.

[2]Note that, with the choice of $K = 0$ and $h = h_0$, $\tilde{\nu}_N(g)$ reduces to the original sample average $\nu_N(g)$.

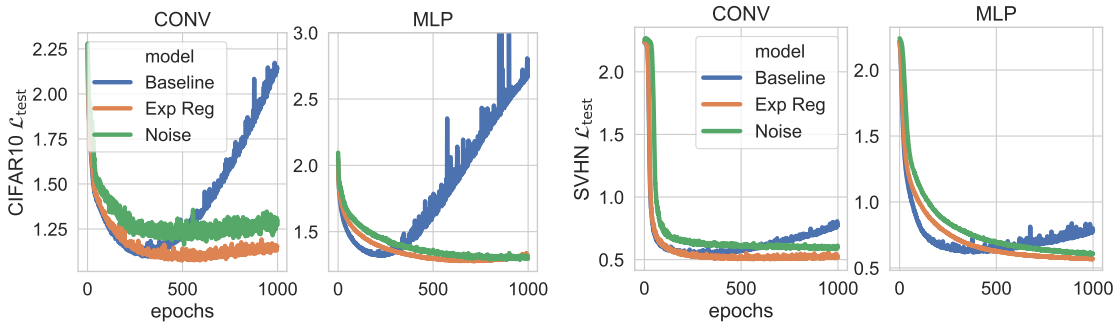


Figure 4.7: We show the test-set loss for SVHN [left] and CIFAR10 [right], for 2-layer convolutional (CONV) and 4-layer MLPs with 512 units per layer trained with the explicit regulariser approximation $R(\cdot)$ of Camuto et al. (2020) (Exp Reg), with additive-GNIs ($\sigma^2 = 0.1$) (Noise), and no regularisation (Baseline).

Theorem 4.6. *Let $\gamma := \alpha - 2 \in (-1, 0)$. Suppose that the assumptions stated in the Appendix hold. Then, the following bound holds almost surely:*

$$|\nu(g) - \lim_{N\to\infty} \tilde{\nu}_N(g)| \leq \frac{\tilde{C}}{4\pi(|\gamma|+2)} \left[|\theta||\gamma| + |\tan(\gamma\pi/2)|\,|\gamma|\right] h + \left((1+\theta)C_0' + (1-\theta)C_0''\right)\frac{1}{hK} + \mathcal{O}(h^2)\,, \tag{4.21}$$

where $\tilde{C}$, C_0', $C_0'' > 0$ are constants.

We note our result extends the case $\alpha = 2$, $\theta = 0$ in Durmus and Moulines (2017) and the case $\alpha \neq 2$, $\theta = 0$ in Şimşekli (2017); whereas we cover the case $\alpha \neq 2$, $\theta \in (-1, 1)$. The right-hand-side of (4.21) contains two main terms. The second term shows that the error increases linearly with decreasing K, indicating that the error can be *arbitrarily large* when $K = 0$, and the gap cannot be controlled without imposing further assumptions on f.

More interestingly, even when K goes to infinity (i.e., the second term vanishes), the first term stays unaffected. Note that for large enough ε, h_0 increases as $\alpha \in (1, 2)$ decreases. In this regime, the first term indicates that the error increases with decreasing α, and an additional error term appears whenever $\theta \neq 0$, which is further amplified with the heaviness of the tails (measured by $|\gamma|$). This outcome provides a theoretical justification to the empirical observations stated in Figures 4.6 and 4.8.

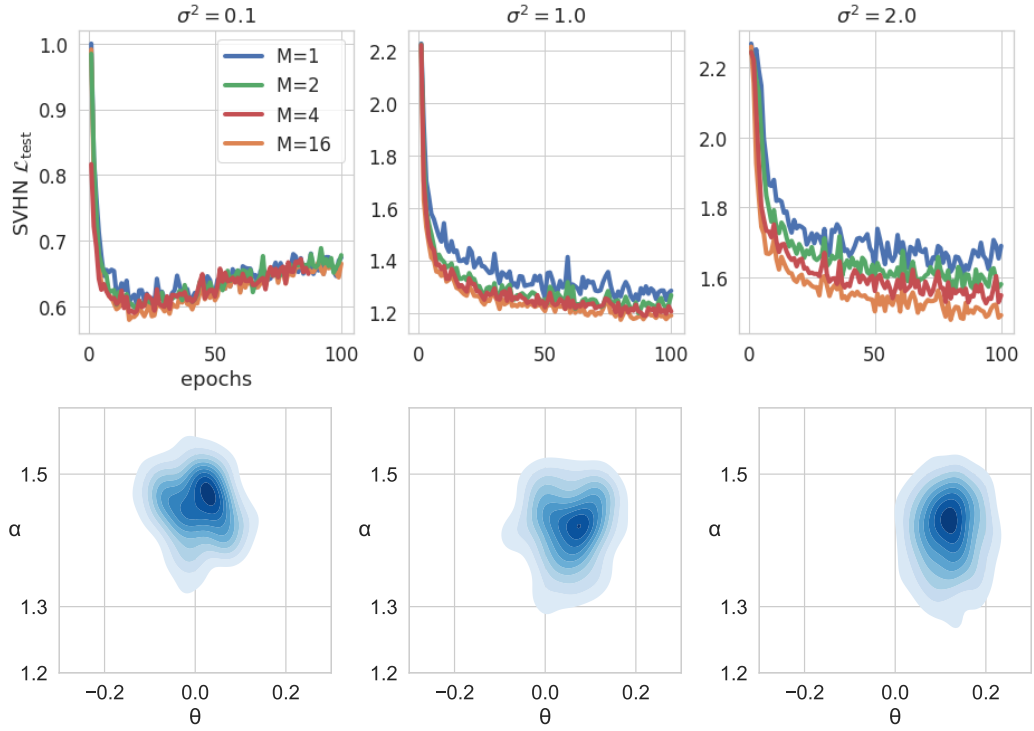


Figure 4.8: [first row] We train 2-dense-layer-256-unit-per-layer ELU networks on the objective $\frac{1}{M}\sum_{m=0}^{M}\widetilde{\mathcal{L}}(\mathcal{B};\mathbf{w},\boldsymbol{\epsilon}_m)$ with a cross-entropy loss (see Appendix B.2) for SVHN. We use *multiplicative* noise of variance σ^2 and batch size of 512. We plot the test-set loss ($\mathcal{L}_{\text{test}}$). [second row] We fit univariate $\mathcal{S}_\alpha$ via maximum likelihood (Nolan, 2001) to $\nabla E_{\mathcal{L}}(\cdot)$ and show KDE plots of parameters' estimates.

4.5 Further Experiments

We have already ascertained that the bias implied by Theorem 4.6 has a visible impact on training performance in Figures 4.4 and 4.5. We have also already shown that the asymmetry and heavy-tails of the implicit-effect gradient noise are responsible for this performance degradation in Figure 4.4: models trained with Gaussian noise on gradients outperform models trained with $\mathcal{S}_\alpha$ gradient noise, and those trained with the implicit effect, *on training data without batching*. We corroborate these findings with experiments *with mini-batching and results on test data*. In Figure 4.7 we use the approximation of the explicit regulariser $R(\mathcal{B};\mathbf{w})$ derived by Camuto et al. (2020) for computational efficiency. Convolutional networks trained with R consistently outperform those trained with GNIs and mini-batching on *held-out data*, supporting that the implicit effect degrades performance. In Figure 4.8, we sample M multiplicative-GNI samples and marginalise out the implicit effect as before. We model the gradients of the implicit effect, $\nabla E_{\mathcal{L}}(\cdot)$, as an $\mathcal{S}_\alpha$

distribution. Empirically, we found that when increasing the variance (σ^2) of the injected noise, the gradient noise $\nabla E_{\mathcal{L}}(\cdot)$ becomes increasingly heavy-tailed and skewed, i.e. α decreases and $|\theta|$ increases, and in tandem larger M models begin to outperform smaller M models on *held-out data*, when trained with mini-batches. These results support that GNIs induce bias in SGD because of the asymmetric heavy-tailed noise they induce on gradient updates.

4.6 Conclusion

Our work lays the foundations for the study of regularisation methods from the perspective of SDEs. We have shown that Gaussian Noise Injections (GNIs), though they inject Gaussian noise in the forward pass, induce asymmetric heavy-tailed noise on gradient updates by way of the implicit effect. By modelling the overall induced noise using an asymmetric α-stable noise, we demonstrate that the stationary distribution of this process gets arbitrarily distant from the so-called Gibbs measure, whose modes exactly match the local minima of the loss function, shedding light on why neural networks trained with GNIs underperform networks trained solely with the explicit effect. Given the deleterious effects of asymmetric gradient noise on gradient descent, extensions of this work could focus on methods that symmetrise gradient noise, stemming from batching or noise injections, so as to limit these negative effects.

Acknowledgements

This research was directly funded by the Alan Turing Institute under Engineering and Physical Sciences Research Council (EPSRC) grant EP/N510129/1. Alexander Camuto was supported by an EPSRC Studentship. Xiaoyu Wang and Lingjiong Zhu are partially supported by the grant NSF DMS-2053454 from the National Science Foundation. Lingjiong Zhu is also grateful to the partial support from a Simons Foundation Collaboration Grant. Mert Gürbüzbalaban's research is supported in part by the grants Office of Naval Research Award Number N00014-21-1-2244, National Science Foundation (NSF) CCF-1814888, NSF DMS-1723085, NSF DMS-2053485. Umut Şimşekli's research

is partly supported by the French government under management of Agence Nationale de la Recherche as part of the “Investissements d’avenir” program, reference ANR-19-P3IA-0001 (PRAIRIE 3IA Institute).

CHAPTER 5

TOWARDS A THEORETICAL UNDERSTANDING OF THE ROBUSTNESS OF VARIATIONAL AUTOENCODERS

This work was written in collaboration with Matthew Willetts, Stephen Roberts, Chris Holmes, and Tom Rainforth.

5.1 Introduction . . . 68

5.2 Background . . . 70

5.2.1 Variational Autoencoders . . . 70

5.2.2 Adversarial Attacks for VAEs . . . 71

5.2.3 Disentangled VAEs . . . 71

5.3 Robustness of VAEs . . . 72

5.4 A Robustness Margin for VAEs . . . 73

5.4.1 r-robustness for VAEs . . . 74

5.4.2 Robustness to distortions in data-space . . . 75

5.5 Characterizing the Margin . . . 76

5.6 Empirical Investigations . . . 79

5.7 Robustness of Disentangled VAEs . . . 81

5.8 Conclusion . . . 83

Abstract

We make inroads into understanding the robustness of Variational Autoencoders (VAEs) to adversarial attacks and other input perturbations. While previous work has developed algorithmic approaches to attacking and defending VAEs, there remains a lack of formalization for what it means for a VAE to be robust. To address this, we develop a novel criterion for robustness in probabilistic models: r-robustness. We then use this to construct the first theoretical results for the robustness of VAEs, deriving margins in the input space for which we can provide guarantees about the resulting reconstruction. Informally, we are able to define a region within which any perturbation will produce a reconstruction that is similar to the original reconstruction. To support our analysis, we show that VAEs trained using disentangling methods not only score well under our robustness metrics, but that the reasons for this can be interpreted through our theoretical results.

5.1 Introduction

Variational Autoencoders (VAEs) (Rezende et al., 2014a; Kingma and Welling, 2014) have been found to be more robust to input perturbations than their deterministic counterparts, particularly those originating from adversarial attacks (Willetts et al., 2020; Schott et al., 2019; Ghosh et al., 2018). This trait has made them useful in protecting downstream tasks (Willetts et al., 2020; Schott et al., 2019; Ghosh et al., 2018).

Nevertheless, they are still not completely impervious to attack (Tabacof et al., 2016; Gondim-Ribeiro et al., 2018a; Kos et al., 2018b): a hypothetical adversary can attack a VAE by applying small input perturbations to invoke meaningful changes in the encoding. Typically this is done by trying to find perturbations which produce reconstructions close to a distinct target datapoint chosen by the adversary, rather than being representative of the original input. Such attacks have been shown to be successful in a wide range of scenarios (Tabacof et al., 2016; Gondim-Ribeiro et al., 2018a; Kos et al., 2018b; Willetts et al., 2020). Recent work has made progress towards defending against them from an empirical and algorithmic perspective (Willetts et al., 2020), by repurposing approaches designed to learn disentangled latent representations (Burgess et al., 2018; Chen et al., 2018; Mathieu et al., 2019).



Figure 5.1: Reconstructions under attack for robust and non-robust VAEs. Each subfigure shows from left to right: the original input, a perturbed input made by an adversarial attack, and the reconstruction of the perturbed input. We show results for VAEs that are robust ($R^r_{\mathcal{X}}(\mathbf{x}) \geq \|\boldsymbol{\delta}_x\|_2$) and non-robust ($R^r_{\mathcal{X}}(\mathbf{x}) < \|\boldsymbol{\delta}_x\|_2$) for a given point $\mathbf{x}$ and adversarially selected perturbation $\boldsymbol{\delta}_x$. We see that the robust VAE reconstructions are visually closer to the original input.

However, a deeper understanding of the mechanisms underpinning the robustness of VAEs and their derivatives is still lacking. Furthermore, there are currently no theoretical foundations for this robustness or even any frameworks or formalizations for exactly what it means for a VAE to be "robust." In other words, what would it mean to have a certifiably–robust VAE? Moreover, are there scenarios where we might be able to provide theoretical guarantees of such robustness?

As a first step to addressing these questions, we develop the first metric with which to evaluate the robustness of VAEs: r-robustness. Informally, for a given input, a VAE is r-robust to a given perturbation if it is more likely that its reconstruction will fall within a ball of radius r around the undistorted maximum likelihood reconstruction, than outside it. The smaller the value of r for which we can confirm r-robustness, the more robust we can guarantee the VAE to be. Through r-robustness, we provide theoretical foundations to understand the source of VAEs' robustness and provide insights into what can cause them to more or less robust.

Using this, we next develop a *margin* of robustness, $R^r_{\mathcal{X}}(\mathbf{x})$, such that the VAE is r-robust to *any* possible perturbations of the input $\mathbf{x}$ within this margin. This, in turn, allows us to provide a notion of a certifiably–robust reconstruction as it forms a guarantee that no attack limited to the margin can reliably undermine it. An example of this is shown in Fig 5.1, where we demonstrate that large $R^r_{\mathcal{X}}(\mathbf{x})$ are associated with model–input pairs that are robust to adversarially generated input perturbations. Analogously to the concept of an adversarial risk (Uesato et al., 2018), $R^r_{\mathcal{X}}(\mathbf{x})$ can further be converted to a metric for the *overall* robustness of a VAE, by taking its expectation over the data generating distribution.

To make inroads towards imposing apriori constraints on a VAE that ensure that it is certifiably robust, we further derive a theoretical bound for $R^r_{\mathcal{X}}(\mathbf{x})$ as a function of the encoder variance and Jacobian. This provides insights into the characteristics of VAEs that contribute to robustness. Building on this result, we show empirically that VAEs with larger encoder variances and smaller Jacobians typically produce larger margins $R^r_{\mathcal{X}}(\mathbf{x})$ and are thus more robust to perturbations. We further demonstrate how these beneficial characteristics can be induced using methods introduced to learn disentangled representations, deriving new results for how these methods can be interpreted.

To summarize, our core contributions are that we first define a robustness metric, r-robustness, that is tailored to probabilistic generative models. We develop a margin $R^r_{\mathcal{X}}(\mathbf{x})$ on a VAE's input space within which it is r-robust to perturbations. Finally, we offer theoretical and empirical analysis—based on $R^r_{\mathcal{X}}(\mathbf{x})$ and existing disentanglement methods—that can aid the construction of robust VAEs.

5.2 Background

5.2.1 Variational Autoencoders

VAEs (Rezende et al., 2014a; Kingma and Welling, 2014), and the models they have inspired (Alemi et al., 2017; Higgins et al., 2017a; Chen et al., 2018; Willetts et al., 2020), are deep latent variable models. Using $\mathbf{x} \in \mathcal{X}$ to denote data and $\mathbf{z} \in \mathcal{Z}$ to denote the latents with associated prior $p(\mathbf{z})$, a VAE simultaneously learns both a forward generative model, $p_\theta(\mathbf{x}|\mathbf{z})$, and an amortised approximate posterior distribution, $q_\phi(\mathbf{z}|\mathbf{x})$, where θ and ϕ correspond to their respective parameters, typically taking the form of neural networks. These are referred to as the decoder and encoder respectively, and a VAE can be thought of as a deep stochastic autoencoder. Under this autoencoder framework, one typically takes the reconstructions as deterministic, corresponding to the mean of the decoder, namely $g_\theta(\mathbf{z}) := \mathbb{E}_{p_\theta(\mathbf{x}|\mathbf{z})}[\mathbf{x}]$, a convention we adopt.

A VAE is trained by maximizing the evidence lower bound (ELBO) $\mathcal{L} = \mathbb{E}_{p_\mathcal{D}(\mathbf{x})}[\mathcal{L}(\mathbf{x})]$, where

$$\mathcal{L}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

and $p_{\mathcal{D}(\mathbf{x})}$ represents the empirical data distribution. The optimization is carried out using stochastic gradient descent with Monte Carlo samples, typically employing the reparameterization trick (Kingma and Welling, 2014). For example, for a Gaussian $q_\phi(\mathbf{z}|\mathbf{x})$, we draw samples as $\mathbf{z} = \boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\eta} \circ \boldsymbol{\sigma}_\phi(\mathbf{x}), \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, where $\circ$ is the element-wise product.

5.2.2 Adversarial Attacks for VAEs

In adversarial settings, an agent is trying to alter the behavior of a model towards a specific goal. This could involve, in the case of classification, adding a very small perturbation to an input so as to alter the model's predicted class. For many deep learning models, small changes to data imperceptible to the human eye, can drastically change a model's output.

Proposed attacks for VAEs aim to produce reconstructions close to a chosen target image by applying small distortions to the input of a VAE (Tabacof et al., 2016; Gondim-Ribeiro et al., 2018b; Kos et al., 2018a). The adversary optimizes this perturbation to minimize some measure of distance between the reconstruction and the target image *or* the distance between the embedding of the distorted image and the embedding of the target image.

5.2.3 Disentangled VAEs

Learning *disentangled* representations (Bengio et al., 2013) involves training a probabilistic generative model in a manner that encourages a one-to-one correspondence between dimensions of the learnt latent space and some interpretable aspect of the data (Higgins et al., 2017a; Alemi et al., 2017; Burgess et al., 2018; Chen et al., 2018; Mathieu et al., 2019). For instance, there might be some axis in latent space encoding 'hair color' for images of people. One such method is the β-VAE (Burgess et al., 2018; Higgins et al., 2017a), which upweights the KL in the ELBO with a penalization factor β:

$$\mathcal{L}_\beta(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \beta \, \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))]$$

Disentangling can be difficult to achieve in practice, requiring careful hyperparameter tuning (Locatello et al., 2019; Mathieu et al., 2019; Rolinek et al., 2019).

Nevertheless, models trained under disentangling objectives have other beneficial properties. In particular, β-TC regularization (Chen et al., 2018), another method proposed for disentangling, has been shown to induce models that are more robust to adversarial attacks (Willetts et al., 2020). The encoders of β-VAEs have also been used as the perceptual part of Deep RL models to create more robust agents (Higgins et al., 2017c). Thus, regardless of the presence of disentangled generative factors, these regularization methods can induce models that are more robust to attack.

5.3 Robustness of VAEs

A Probabilistic Metric of Robustness Deep learning models can be brittle. Some of the most sophisticated deep learning classifiers can be broken by simply adding small pertubations to their inputs (Szegedy et al., 2014; Shamir et al., 2019; Goodfellow et al., 2015; Papernot et al., 2016; Moosavi-Dezfooli et al., 2016). Here, perturbations that would not fool a human break neural network predictions. A model's weakness to such perturbations is called its *sensitivity*. For classifiers, we can straightforwardly define an associated *sensitivity margin*: it is the radius of the largest metric ball centered on an input $\mathbf{x}$ for which a classifier's original prediction holds for all possible perturbations within that ball.

Defining such a margin for VAEs is conceptually more difficult as, in general, the reconstructions are continuous rather than discrete. To put it another way, there is no step-change in VAE reconstructions that is akin to a change of a predicted class in classifiers; *any* perturbation in the input space will result in a change in the VAE output. To complicate matters further, a VAE's latent space is stochastic: the same input can result in different reconstructions.

As a first step to deriving robustness margins for VAEs, we now introduce a criterion for measuring robustness in probabilistic models: r-robustness. We start by presenting it in the general setting, before linking it to the specific case of VAEs.

Definition 5.1. *A model, f, operating on a point $\mathbf{x}$, that outputs a continuous random*

variable is r-robust for $r \in \mathbb{R}^+$*, to a perturbation* $\boldsymbol{\delta}$ *and for an arbitrary norm* $||\cdot||$ *iff*

$$p(||f(\mathbf{x}+\boldsymbol{\delta}) - f(\mathbf{x})|| \leq r) > p(||f(\mathbf{x}+\boldsymbol{\delta}) - f(\mathbf{x})|| > r).$$

We will assume from now on that the norm is taken to be the 2-norm $||\cdot||_2$, such that r-robustness determines a bound for which changes in the output $f(\mathbf{x})$ induced by the perturbation $\boldsymbol{\delta}$ are more likely to fall within the hyper-sphere of radius r, than not. As r decreases, the criterion for model robustness becomes stricter. We note that r-robustness can be viewed as a probabilistic analog to the criterion for regression models presented by Nguyen and Raff (2019). We also note that r-robustness can be generalized to $p(||f(\mathbf{x}+\boldsymbol{\delta}) - f(\mathbf{x})||_2 \leq r) > m$, where $1-m$ is the allowable risk (with $m = 0.5$ in Definition 5.1).

Because this criterion is applicable to probabilistic models with continuous output spaces, it is directly relevant for ascertaining robustness in VAEs. By considering the smallest r for which the criterion holds, we can think of it as a metric that provides a probabilistic measure of the *extent* to which outputs are altered given a corrupted input: the smaller the value of r for which we can confirm r-robustness, the more robust the model.

5.4 A Robustness Margin for VAEs

We want to define a margin in a VAE's input space for which it is robust to perturbations of a given input. Perturbations that fall within this margin should not break our criterion for robustness. Formally, we want a margin in $\mathcal{X}$, $R^r_{\mathcal{X}}(\mathbf{x})$, for which any distorted input $\mathbf{x}+\boldsymbol{\delta}_x$, where $||\boldsymbol{\delta}_x||_2 < R^r_{\mathcal{X}}(\mathbf{x})$ is the perturbation, satisfies r-robustness when reconstructed.

However, to consider the robustness of VAEs, we must not only take into account the perturbation $\boldsymbol{\delta}_x$, but also the stochasticity of encoder. We can think of the decoder as taking in noisy inputs because of this stochasticity. Naturally, this noise can itself potentially cause issues in the robustness of VAE: if the level of noise is too high, we will not achieve reliable reconstructions even without perturbing the original inputs. As such, before even considering perturbations, we first need to adapt our r-robustness framework

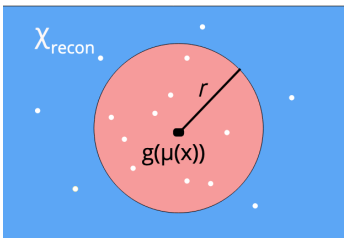


Figure 5.2: Illustration of r-robustness in a VAE. White dots represent possible reconstructions, with the diversity originating from the encoder stochasticity. For r-robustness to hold, the probability of our reconstruction falling within the red area—a hypersphere of radius r centered on $g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))$—needs to be greater than or equal to the probability of falling outside.

to deal with this stochasticity.

5.4.1 r-robustness for VAEs

Given an input $\mathbf{x}$, r-robustness dictates that we want to define some region in the reconstruction space, $\mathcal{X}_{\text{recon}}$, within which most of the decoded samples from the latent embedding $\mathbf{z}$ will fall. We will assume that the encoder is a Gaussian as this is standard practice. Denoting $g_\theta(\mathbf{z})$ as the deterministic mapping induced by the VAE's decoder network and $\boldsymbol{\mu}_\phi(\mathbf{x})$ as the mean embedding of the encoder, we can define $g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))$ to be the "maximum likelihood" reconstruction, noting this is a deterministic function. Our aim is now to find a hyper-sphere of radius r centered on $g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))$ within which most of the possible VAE outputs for a given point $\mathbf{x}$ lie. Larger r are indicative of a greater variance in the encoding process, and as such are likely to be associated with poorer quality reconstructions.

Denoting $\boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ as the reparameterized stochasticity of the encoder, we define the distance from the maximum likelihood reconstruction, induced by this sampling as

$$\Delta(\mathbf{x}) = g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\eta} \circ \boldsymbol{\sigma}_\phi(\mathbf{x})) - g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x})) \tag{5.1}$$

Using this, we see that a VAE is r-robust to the stochasticity of the encoder iff (see also Fig 5.2)

$$p(||\Delta(\mathbf{x})||_2 \leq r) > p(||\Delta(\mathbf{x})||_2 > r). \tag{5.2}$$

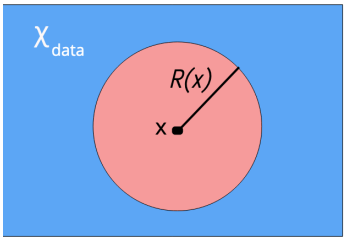


Figure 5.3: Illustration of the margin $R^r_{\mathcal{X}}(\mathbf{x})$, which is defined in the **input** space $\mathcal{X}$. Red represents represents the subspace where the model is r-robust, such that $p(||\Delta(\mathbf{x}, \boldsymbol{\delta}_x)||_2 \leq r) > p(||\Delta(\mathbf{x}, \boldsymbol{\delta}_x)||_2 > r)$ holds for all $\mathbf{x} + \boldsymbol{\delta}_x$ falling in this region, that is all $\boldsymbol{\delta}_x : \|\boldsymbol{\delta}_x\|_2 \leq R^r_{\mathcal{X}}(\mathbf{x})$.

Informally, we want it to be more probable for reconstructions to fall within this radius r than not.

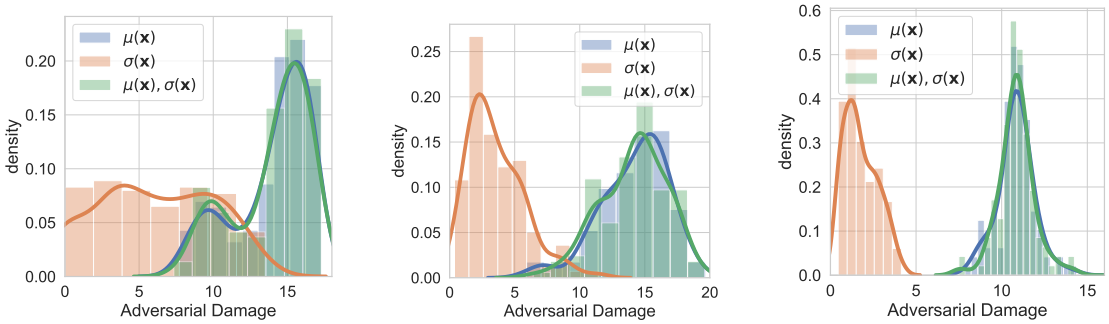
5.4.2 Robustness to distortions in data-space

Given that we have established conditions for r in Eq (5.2) that take into account latent space sampling, we can now return to our original objective, which was to determine a margin in the data-space $\mathcal{X}$ for which a VAE is robust to perturbations on its input. Recall that this implicitly means that we want to define a bound for robustness given two sources of perturbations: the stochasticity of the encoder, and a hypothetical input perturbation $\boldsymbol{\delta}_x$.

For simplicity of analysis, we consider the case where the perturbation is applied only to the encoder mean input and not the encoder variance input, noting that the latter is typically stable across inputs and so is less of a concern. In Fig 5.4 we demonstrate that adversarial attacks on VAE encoders are dominated by the perturbation to the embedding mean that is induced, thereby justifying this assumption. We note also that one can usually also simply fix the encoder variance to a constant for all datapoints without incurring substantial performance drops (Ghosh et al., 2020), thereby providing a means to ensure this assumption holds exactly if needed.

We define the distance from the maximum likelihood reconstruction, $g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))$, induced by the stochasticity of the encoder *and* an input perturbation $\boldsymbol{\delta}_x$ as

$$\Delta(\mathbf{x}, \boldsymbol{\delta}_x) = g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x} + \boldsymbol{\delta}_x) + \boldsymbol{\eta}\sigma_\phi(\mathbf{x})) - g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x})).$$



(a) MNIST (b) fMNIST (c) CIFAR10

Figure 5.4: *Maximum damage* adversarial attacks (see Eq (5.5)) on multiple VAEs trained on MNIST (a), fashion-MNIST (b), and CIFAR10 (c). We attack 25 datapoints for each VAE and propagate the attacks to the encoder mean ($\mu(\mathbf{x})$), the encoder standard deviation ($\sigma(\mathbf{x})$), or both ($\mu(\mathbf{x}), \sigma(\mathbf{x})$). Attack norms are capped to 10. Shown are distribution plots of the adversarial damage, i.e. the L_2 distance between the reconstruction resulting from the attack and the maximum likelihood reconstruction $g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))$. Clearly attacks on $\mu(\mathbf{x})$ are more harmful than on $\sigma(\mathbf{x})$, and most of the damage from attacks on both $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$ stems from the attack on $\mu(\mathbf{x})$.

We can now define the condition for which r-robustness is satisfied on the VAE output given the two sources of perturbation as

$$||\boldsymbol{\delta}_x||_2 < R^r_{\mathcal{X}}(\mathbf{x}) \;\Leftrightarrow\; p(||\Delta(\mathbf{x}, \boldsymbol{\delta}_x)||_2 \leq r) > 0.5 \tag{5.3}$$

Thus, $R^r_{\mathcal{X}}(\mathbf{x})$ is the margin of robustness of the VAE such that $\forall \boldsymbol{\delta}_x \;:\; ||\boldsymbol{\delta}_x||_2 < R^r_{\mathcal{X}}(\mathbf{x})$, $\mathbf{x} + \boldsymbol{\delta}_x$ is more likely than not to be reconstructed within a radius r of the maximum likelihood reconstruction $g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))$. A high level illustration of this is given in Fig 5.3, and Fig 5.5 shows a simple empirical demonstration of how $R^r_{\mathcal{X}}(\mathbf{x})$ relates to the probability of producing a good reconstruction under random input perturbations.

We note that, analogously to the concept of an adversarial risk (Uesato et al., 2018), $R^r_{\mathcal{X}}(\mathbf{x})$ can further be converted to a metric for the *overall* robustness of a VAE, by taking its expectation over the data generating distribution, namely $R^r_{\mathcal{X}} = \mathbb{E}_{p_{\mathcal{D}}(\mathbf{x})}\left[R^r_{\mathcal{X}}(\mathbf{x})\right]$.

5.5 Characterizing the Margin

Given this definition, we now wish to try and characterize $R^r_{\mathcal{X}}(\mathbf{x})$. In particular, we would like to understand what characteristics of the VAE are likely to make it relatively larger

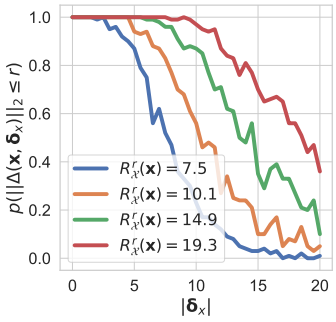


Figure 5.5: $R^r_{\mathcal{X}}(\mathbf{x})$ for four VAEs of varying robustness trained on MNIST. We fix the input $\mathbf{x}$ and perturbation direction $\boldsymbol{\delta}_x/\|\boldsymbol{\delta}_x\|_2$, but vary the perturbation size $\|\boldsymbol{\delta}_x\|_2$. We assess the proportion of samples which fall within $r = 4$ of the maximum likelihood reconstruction.

or smaller. Ideally, we also want to establish scenarios where we might be able to provide guarantees of a minimum size for $R^r_{\mathcal{X}}(\mathbf{x})$, such that we might be able to make inroads into how one might apriori construct a certifiably-robust VAE.

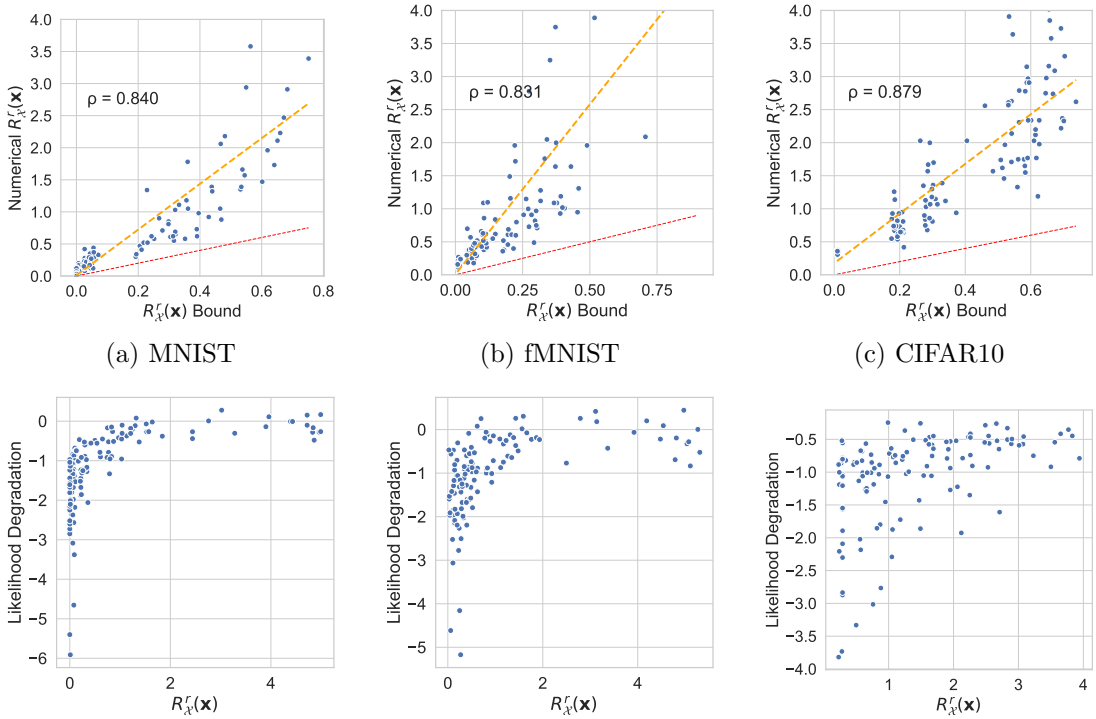
A perturbation in $\mathcal{X}$, $\boldsymbol{\delta}_x$, induces a perturbation in $\mathcal{Z}$, $\boldsymbol{\delta}_z$. To determine the margins for robustness in $\mathcal{X}$, we first apply the Neyman-Pearson lemma (Neyman and Pearson, 1933; Cohen et al., 2019), assuming a "worst-case" decoder. This decoder has subspaces in $\mathcal{Z}$, where it is is either robust or non-robust, that are divided by a boundary that is normal to both the induced perturbation $\boldsymbol{\delta}_z$ and to the dimension of minimal variance in $\mathcal{Z}$, $\min_i \boldsymbol{\sigma}_\phi(\mathbf{x})_i$. We then determine the minimum perturbation norm in $\mathcal{X}$ which induces a perturbation in $\mathcal{Z}$ that crosses this boundary.

Theorem 5.1. *Consider a VAE with a diagonal-variance Gaussian encoder, an input* $\mathbf{x}$*, and an output margin* $r \in \mathbb{R}$ *such that the VAE is* r*-robust to the stochasticity of the encoder when the* $\mathbf{x}$ *is unperturbed as per* (5.2)*. Assuming standard regularity assumptions (discussed in the proof) hold for* $\boldsymbol{\mu}_\phi(\mathbf{x})$*, then*

$$R^r_{\mathcal{X}}(\mathbf{x}) \geq \frac{(\min_i \boldsymbol{\sigma}_\phi(\mathbf{x})_i)\Phi^{-1}(p(||\Delta(\mathbf{x})||_2 \leq r))}{||\mathbf{J}^{\mu}_{\phi}(\mathbf{x})||_F} + \mathcal{O}(\boldsymbol{\varepsilon}) \tag{5.4}$$

where $\mathcal{O}(\boldsymbol{\varepsilon})$ *represents higher order dominated terms that disappear in the limit of small perturbations,* Φ^{-1} *is the probit function,* $\mathbf{J}^{\mu}_{\phi}(\mathbf{x})_{i,j} = \partial \boldsymbol{\mu}_\phi(\mathbf{x})_i / \partial \mathbf{x}_j$ *is the Jacobian of* $\boldsymbol{\mu}_\phi(\mathbf{x})$*, and* $||\cdot||_F$ *is the Frobenius norm.*

The proof is provided in Appendix C.2. This bound is based on a first order approximation of $\boldsymbol{\mu}_\phi(\mathbf{x}+\boldsymbol{\delta}_x)$ around the original input $\mathbf{x}$; the impact of $\mathcal{O}(\boldsymbol{\varepsilon})$ thus depends on how



(a) MNIST (b) fMNIST (c) CIFAR10

(d) MNIST (e) fMNIST (f) CIFAR10

Figure 5.6: (a-c) show the empirically estimated $R^r_{\mathcal{X}}(\mathbf{x})$ against the bound for $R^r_{\mathcal{X}}(\mathbf{x})$ defined in Theorem 1, ignoring higher order terms. Each dot represents a network–input pair, with 5 separately trained networks and 25 distinct inputs considered. We show the line of best fit (in orange), the correlation coefficient ρ, and the line $y = x$ (in red) representing the theoretical bound itself. (d-f) show the relative log likelihood degradation resulting from a 'maximum-damage' adversarial attack against the numerically estimated $R^r_{\mathcal{X}}(\mathbf{x})$ for these same VAEs and inputs (see Section 5.6).

well this approximation holds. As such, the result is particularly applicable to networks with piecewise linear activation functions such as the ReLU, which are locally linear and are among the most widely used activation functions. For these activation functions this bound is locally exact: $\mathcal{O}(\boldsymbol{\varepsilon})$ is exactly zero if the size of the bound is smaller than what is required to go outside the locally linear region.

This gives us margins for which VAEs are certifiably robust, up to first order expansions, to adversarial perturbations on their inputs; they have similar forms to the sensitivity margins for classifiers defined by Sokolić et al. (2017); Jakubovitz and Giryes (2018) in that both scale *inversely* with the network Jacobian. More generally, these results provide insights into the features which lead to robust VAEs. As shown in Figure 5.6, the bound seems to be relatively tight in practice, even when attacking both the encoder mean *and* variance. It also has a near-linear relationship with the empirically estimated robustness,

such that it forms a powerful and convenient robustness metric in its own right.

Examining the bound, we see that for a given r, $R^r_{\mathcal{X}}(\mathbf{x})$ increases as the stochasticity of the encoder, i.e $\boldsymbol{\sigma}_\phi(\mathbf{x})$, increases, provided that this does not overly affect $\Phi^{-1}(p(||\Delta(\mathbf{x})||_2 \leq r))$ (see below). As $\boldsymbol{\sigma}_\phi(\mathbf{x})$ tends to 0 we recover the deterministic setting, which confers no additional protection to attack and as $\boldsymbol{\sigma}_\phi(\mathbf{x})$ increases we obtain increased protection. However, $\boldsymbol{\sigma}_\phi(\mathbf{x})$ can also have a knock–on effect on $\Phi^{-1}(p(||\Delta(\mathbf{x})||_2 \leq r))$. When $\boldsymbol{\sigma}_\phi(\mathbf{x})$ is small, this knock–on effect will typically be small relative to the direct effect of changing $\boldsymbol{\sigma}_\phi(\mathbf{x})$, but as it becomes large there is always a point where this knock–on effect will take over. Namely, our reconstructions will become increasingly poor and $\Phi^{-1}(p(||\Delta(\mathbf{x})||_2 \leq r))$ will eventually become negative, such that r-robustness does not hold even without perturbation. We can quantify this by noting that there is always a minimum r for r-robustness to be satisfied in (5.2). We derive a bound characterizing the minimum r for which we can confirm robustness in Appendix C.1.

5.6 Empirical Investigations

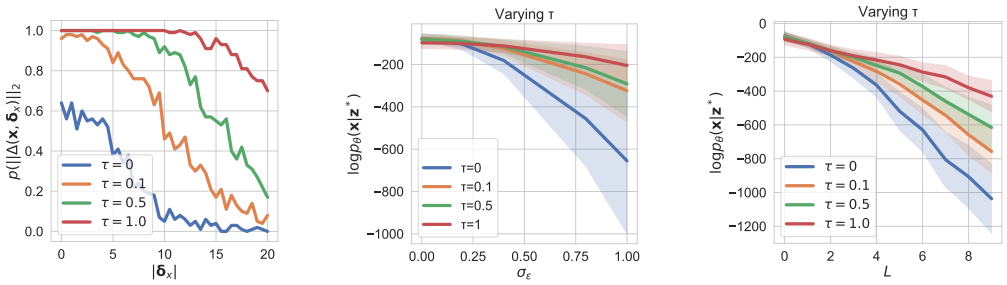


Figure 5.7: Ablation study on the bounds defined by Theorem 5.1. We train models on MNIST with $\sigma_\phi(\mathbf{x})$ offset by a constant $\tau \in [0, 0.1, 0.5, 1]$. [Left] probability that reconstructions in $\mathcal{X}_{\text{recon}}$ fall within a radius $r = 4$ centered on the 'maximum likelihood' reconstruction, $p(||\Delta(\mathbf{x}, \boldsymbol{\delta}_x)||_2 \leq r)$, as a function of $|\boldsymbol{\delta}_x|$, the magnitude of perturbations. $R^r_{\mathcal{X}}(\mathbf{x})$ is the radius $|\boldsymbol{\delta}_x|$ for which $p(||\Delta(\mathbf{x}, \boldsymbol{\delta}_x)||_2 \leq 4) > 0.5$ and clearly increases with τ. [Center] we add noise $\sim \mathcal{N}(0, \sigma^2_\epsilon)$ to a point $\mathbf{x}$ forming a noisy $\mathbf{x}^*$ and $\mathbf{z}^*$, and measure the likelihood of the original point $\mathbf{x}$ under this noisy embedding. [Right] we show the same plot where the perturbations are *maximum damage* attacks, Eq (5.5), where L is the maximum allowed magnitude of the attack distortion. Large τ VAEs have high likelihoods for the original point $\mathbf{x}$ as L and σ^2_ϵ increase: they are robust to attack and effective denoising models. Confidence intervals are the standard deviations of values over the entire MNIST dataset.

We now consider a series of empirical investigations to back up our frameworks and theoretical results. We start by assessing whether the concept of r-robustness corresponds to more commonly used measures of model robustness. Here we estimate $R^r_{\mathcal{X}}(\mathbf{x})$ numerically as in Appendix C.4 and as demonstrated in Fig 5.5. Using these empirical estimations, we establish connections between r-robustness and other performance metrics during adversarial attack, confirming that larger $R^r_{\mathcal{X}}(\mathbf{x})$ correspond to model–input pairs that are more robust to adversarial attacks.

r-robustness and Adversarial Settings We begin by evaluating our metrics in adversarial settings. We want to find the most damaging perturbations $\boldsymbol{\delta}_x$ that challenge the robustness metrics we have derived. We consider an adversary trying to distort the input data to maximally disrupt a VAE's reconstruction. Our adversary maximizes, wrt $\boldsymbol{\delta}_x$, the distance between the VAE reconstruction and the original datapoint $\mathbf{x}$, a novel adversarial attack we call *maximum damage*. We attack the encoder mean *and* variance:

$$\boldsymbol{\delta}^*_x = \arg\max_{\boldsymbol{\delta}_x} \left(||g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}+\boldsymbol{\delta}_x) + \boldsymbol{\eta}\sigma_\phi(\mathbf{x}+\boldsymbol{\delta}_x)) - g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))||_2 \right), \ \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{5.5}$$

We evaluate the success of an attack as follows. Given an embedding $\mathbf{z}^*$ formed from the mean encoding of $\mathbf{x} + \boldsymbol{\delta}_x$, we measure the likelihood of the original point $\mathbf{x}$ and quantify the degradation in model performance as the relative log likelihood degradation $(|(\log p(\mathbf{x}|\mathbf{z}^*) - \log p(\mathbf{x}|\mathbf{z}))|/\log p(\mathbf{x}|\mathbf{z}))$, where $\mathbf{z}$ is the embedding of $\mathbf{x}$. Fig 5.6 (d-f) shows that as $R^r_{\mathcal{X}}(\mathbf{x})$ increases this degradation lessens, indicating less damaging attacks. As such, larger margins for r-robustness correspond to models that are more robust to attack.

Evaluating the derived bounds Using Theorem 5.1, we can gain insights into which characteristics of a VAE contribute to robustness. The encoder variance plays a prominent role and is a parameter that is easy to control. The encoder Jacobian is also present, but we found that controlling such values directly can be difficult. Penalizing the norm of this Jacobian in the VAE training objective degrades VAE generative performance, making it difficult to compare models. As such we restrict our experiments to varying the encoder

variance. We do so by training models that have $\boldsymbol{\sigma}_\phi(\mathbf{x})$ offset by a constant τ, such that we artificially increase the encoder variance minimum. In Fig 5.7 we show that as τ increases, the numerically estimated $R^r_{\mathcal{X}}(\mathbf{x})$ also increases, supporting our claim that models with larger encoder variances have larger margins of robustness. This figure also shows that likelihood of reconstructing the original input x increases as τ increases in both an adversarial attack setting and a noisy perturbation setting. We thus see that larger τ also provides more effective denoising properties.

5.7 Robustness of Disentangled VAEs

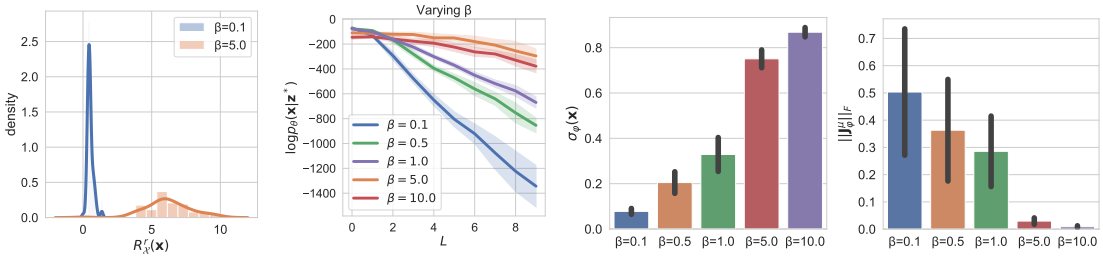


(a) $\beta = 0.1$ (b) $\beta = 1$ (c) $\beta = 10$

Figure 5.8: For β-VAEs trained with $\beta \in \{0.1, 1, 10\}$ we show in consecutive rows first the original data point, a perturbed version made by maximum damage adversarial attacks, and then the reconstruction given by the model. As β increases the models become more robust to attack.

We now apply our analysis to disentangling methods, which have empirically been shown to be more robust to adversarial attacks and noisy data (Willetts et al., 2020). First, we demonstrate this visually in Fig 5.8 where we see that β-VAEs are more resilient to attack as β increases, and thus implicitly latent space overlap increases (Mathieu et al., 2019). Second, we provide analysis to show that disentangling methods induce models with smaller encoder Jacobian norms *and* larger posterior variances, implying that they have larger margins $R^r_{\mathcal{X}}(\mathbf{x})$ by Theorem 5.1, a result we confirm empirically.

Disentangling increases encoder variance Empirically, increasing $\beta > 1$ in a β-VAE increases the variance of the trained encoder, saturating at the variance of the prior for large β (Mathieu et al., 2019; Locatello et al., 2019). We can shed light on this behavior by finding the optimum forms of the posterior distribution under these objective functions using calculus of variations. We find that the optimal posterior has the form of a tempered or fractional posterior (Holmes and Walker, 2017; Wenzel et al., 2020; Miller and Dunson, 2019), with an exponent $1/\beta$ on the likelihood:



(a) Sampled $R^r_{\mathcal{X}}(\mathbf{x})$ (b) Adv Attack (c) Encoder Variance (d) Encoder Jacobian

Figure 5.9: (a) distribution of the numerically estimated $R^r_{\mathcal{X}}(\mathbf{x})$ ($m = 0.5$) across the MNIST dataset. We see that $R^r_{\mathcal{X}}(\mathbf{x})$ increases dataset-wide for larger β. (b) likelihood of the original input given a maximum damage adversarial attack as in Eq (5.5). L is the maximum allowed norm of the attack. Large β models retain high likelihoods even for large L, meaning they are robust to attack. (c) and (d) show that the encoder variance increases and the encoder Jacobian norm ($||\mathbf{J}^\mu_\phi(\mathbf{x})||_F$) decreases as β increases, supporting our analysis that the changes in these values underpin the robustness observed. Confidence intervals for all plots are the standard deviation of values over the entire MNIST dataset. See Appendix C.6 for similar experiments on other datasets.

Theorem 5.2. *For a β-VAE, the optimum posterior is:*

$$q_\phi(\mathbf{z}|\mathbf{x}) \propto p(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})^{1/\beta}$$

The proof is given in Appendix C.3.

This result gives the optimal posterior as a function of β. It also tells us the β-VAE's optimal posterior in the limit of large β is the prior, as we would expect. Because the prior variance is naturally larger than that of the encoder, the encoder variance increases with β.

Disentangling penalizes Jacobian norm Assuming a Gaussian $p_\theta(\mathbf{x}|\mathbf{z})$, an encoder covariance optimal to first order, and activation functions that are piecewise-linear, the β–VAE objective can be approximated as (Kumar and Poole, 2020)

$$\min_{\phi,\theta} \frac{1}{2}||\mathbf{x} - g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))||^2 + \frac{\beta}{2}||\mathbf{J}^\mu_\phi(\mathbf{x})\mathbf{x}||^2_F + \frac{\beta}{2}\log|\mathbf{I} + \frac{1}{\beta}\mathbf{J}_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))\mathbf{J}^T_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))|, \tag{5.6}$$

As β increases $||\mathbf{J}^\mu_\phi(\mathbf{x})||^2_F$ is more penalised and we expect to learn encoders with smaller Jacobians.

Taking these results together, we expect two things to occur as β increases: the encoder

variance should increase by Theorem 5.2 and the norm of the encoder Jacobian should decrease. We confirm this empirically in Fig 5.9(c,d). By Theorem 5.1 these two effects of increasing β should increase $R^r_{\mathcal{X}}(\mathbf{x})$ in tandem. In Fig 5.9(a) we confirm that the numerical estimate for $R^r_{\mathcal{X}}(\mathbf{x})$ increases *dataset-wide* for large β, that is we get a larger value for $R^r_{\mathcal{X}}$, or metric for the overall robustness of the VAE. In Appendix C.6, we also show that the distribution of the *bound* for $R^r_{\mathcal{X}}(\mathbf{x})$ from Theorem 5.1 increases dataset-wide with increasing β. In both cases it is noticeable that $R^r_{\mathcal{X}}(\mathbf{x})$ is quite a well-behaved distribution, with reasonably low variance and skew. This suggests that $R^r_{\mathcal{X}}$ can be reliably estimated in practice.

In Fig 5.9b we further show that these larger $R^r_{\mathcal{X}}$ values translate into larger likelihoods of the original input under adversarial attack, while in Appendix C.6 we confirm that model sensitivity to noise is improved for larger β. We note, however, that having a β that is too large will completely undermine reconstructions (Higgins et al., 2017a; Chen et al., 2018) and lead to VAEs that are never robust because we cannot confirm r-robustness, even without input perturbations (Eq (5.2)). See Appendix C.4.2 for results on this.

5.8 Conclusion

We have defined a novel robustness metric tailored to probabilistic generative models, r-robustness, which can be used to assess the robustness of VAEs to adversarial attack. We defined a margin on a VAE's input space within which it is r-robust to perturbations and show that small norms of the encoder Jacobian and larger encoder variances are core contributors to robustness. Further, we offered theoretical and empirical analysis based on this margin, demonstrating that existing disentangling methods increase robustness by altering the optimal encoder variance and the norm of the encoder Jacobian.

Acknowledgments

This research was directly funded by the Alan Turing Institute under Engineering and Physical Sciences Research Council (EPSRC) grant EP/N510129/1. AC was supported

by an EPSRC Studentship. MW was supported by EPSRC grant EP/G03706X/1. SR gratefully acknowledges support from the UK Royal Academy of Engineering and the Oxford-Man Institute. CH was supported by the Medical Research Council, the Engineering and Physical Sciences Research Council, Health Data Research UK, and the Li Ka Shing Foundation

We thank Tomas Lazauskas, Jim Madge and Oscar Giles from the Alan Turing Institute's Research Engineering team for their help and support.

CHAPTER 6

VARIATIONAL AUTOENCODERS: A HARMONIC PERSPECTIVE

This work was written in collaboration with Matthew Willetts.

6.1 Introduction 86

6.2 Background 89

6.3 Gaussian Spaces and VAEs 91

6.3.1 Gaussian Spaces and VAE decoders 94

6.3.2 Gaussian Spaces and VAE encoders 96

6.3.3 The KL and frequency content 99

6.3.4 VAE Lipschitzness 100

6.3.5 Potential Limitations 101

6.4 Conclusion 101

Abstract

In this work we study Variational Autoencoders (VAEs) from the perspective of harmonic analysis. By viewing a VAE's latent space as a Gaussian Space, a variety of measure space, we derive a series of results that show that the encoder variance of a VAE controls the frequency content of the functions parameterised by the VAE encoder and decoder neural networks. In particular we demonstrate that larger encoder variances reduce the high frequency content of these functions. Our analysis allows us to show that increasing this variance effectively induces a soft Lipschitz constraint on the decoder network of a VAE, which is a core contributor to the adversarial robustness of VAEs. We further demonstrate that adding Gaussian noise to the input of a VAE allows us to more finely control the frequency content and the Lipschitz constant of the VAE encoder networks. Finally, we show that the KL term of the VAE loss serves as single point of action for modulating the frequency content of both encoder and decoder networks; whereby upweighting this term decreases the high-frequency content of both networks. To support our theoretical analysis we run experiments using VAEs with small fully-connected neural networks and with larger convolutional networks, demonstrating empirically that our theory holds for a variety of neural network architectures.

6.1 Introduction

Variational autoencoders (VAEs) are deep latent variable models that typically use Gaussian priors and Gaussian posteriors in their latent spaces (Rezende et al., 2014b; Kingma and Welling, 2014). VAEs have become a work-horse method in modern machine learning, but still their theoretical properties are not fully understood. In particular, we do not yet fully understand the regularising effect of latent space sampling during training.

While the effect of the latent space sampling (a.k.a latent noise) on VAEs has been studied from an information-theoretic standpoint (Shu et al., 2018) and through Taylor analysis (Kumar and Poole, 2020), here we take a different tack and study the impact that latent Gaussian samples have on the *harmonic properties* of the underlying neural networks used to implement the model.

Related to our inquiry, is the study of the effects of *Gaussian noise injections* on neu-

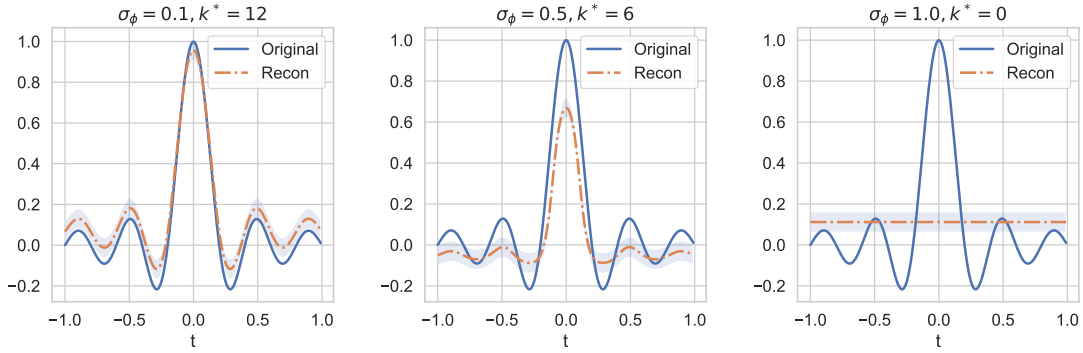


Figure 6.1: We plot the reconstructed function of a VAE, with the Sigmoid activation function networks (3 dense layer each with 256 units), trained on the function $\text{sinc}(5t)$, $t \in [-1, 1]$. We use a one dimensional latent space $\mathcal{Z}$ with a fixed encoder variance $\sigma_\phi \in [0.1, 0.5, 1.0]$. As σ_ϕ increases we can see qualitatively that the frequency of the reconstructed function increasingly skews towards lower frequencies. We fit polynomial regression to the reconstructed function, picking the optimal polynomial degree k^* via cross-validation on 10 randomly chosen train-test splits. As σ_ϕ increases, the lower frequency content of the decoder function corresponds, as one might expect, to a lower optimal polynomial degree k^*.

ral networks trained on supervised tasks (Yin et al., 2019; Camuto et al., 2020, 2021a). Adding (standard) Gaussian noise to the input layer has long been known to induce regularisation (Webb, 1994; Bishop, 1995; Burger and Neubauer, 2003), and recently has been used to induce robustness to adversarial attacks (Cohen et al., 2019). Recently, by studying the effects of these Gaussian noise injections from a functional analysis perspective, (Camuto et al., 2020) shows that these injections effectively penalise functions which learn high frequency components in Fourier-space. These ideas form the starting point for our work.

By considering the latent space of a VAE as a measure space, equipped with a *Gaussian measure*, we can consider the decoder of the VAE as being a member of a *Gaussian Space*, a type of L_2 function space equipped with the Gaussian measure. In our analysis, we view the latent variable's posterior, a Gaussian, as being the Gaussian measure with which the latent space is equipped on a per-datapoint basis. This posterior is located around a particular *location*, the mean, with a particular *scale*, the (often-diagonal) standard deviation.

Previous analysis in Gaussian spaces has been done for spaces equipped with the standard Gaussian measure (zero mean and unit variance). To directly apply the theory

of Gaussian spaces to VAEs, we extend the theory of these spaces for standard Gaussian measures to be applicable to *general* Gaussian spaces with arbitrary mean and covariance.

Using this newly developed theory we consider a basis-expansion of the decoder of a VAE in terms of the eigenfunctions of a general Gaussian space, the *Hermite polynomials*. This then enables us to see how higher degrees of these basis functions, these polynomials, are implicitly more-strongly penalised as the variance of the underlying Gaussian space increases, leading the decoder of VAE to preferentially learn functions that can be represented as lower degree polynomials. As higher-order polynomials are naturally associated with higher frequency content in the Fourier domain, this means that a higher variance Gaussian measure in the latent space necessarily leads to decoders with lower-frequency functional representations.

Further, the encoder, which in our analysis parameterises the Gaussian measure of the latent space, is also affected by changes in the latent measure's variance. By studying the Fourier transform of Gaussian measures we can show very simply that larger-variance measures have less high frequency content in the Fourier domain. This implies that encoders that learn high-variance posteriors have lower-frequency functional representations.

To modulate the frequency content of the *individual networks* that constitute the VAE encoder, we study the effect of adding Gaussian noise to VAE inputs during training. This noising operation effectively turns the input space into a Gaussian space on a per-datapoint basis. In the context of generative models, this process is reminiscent both of Spread Divergences (Zhang et al., 2020b) and dequantisation (Dinh et al., 2017; Salimans et al., 2017; Ho et al., 2019). As we demonstrate both theoretically and empirically, these noise additions enable finer-grained control over the harmonic content of the encoder mean network.

Finally, using this method of harmonic analysis, we extend Nash's Poincaré inequality to general Gaussian spaces. This shows that Gaussian spaces implicitly induce a soft constraint on the Lipschitz constant of their constituent function. In the VAE setting, we use these results to show that the Lipschitz constant of the VAE decoder decreases as the posterior variance increases. Similarly, for the encoder we show that as the variance of Gaussian noise on the VAE inputs increases, the Lipschitz constant of the encoder mean

network decreases.

This result links our work both to recent advances in the Lipschitz penalisation and regularisation of deep generative models (Adler and Lunz, 2018; Terjék, 2020) and more generally to the adversarial robustness literature where control of the Lipschitz constant improves the robustness of models (Gouk et al., 2018; Yang et al., 2020; Hein and Andriushchenko, 2017; Tsuzuku et al., 2018). Our novel theoretical viewpoint on VAEs unifies the two emerging strands in the empirical and theoretical study of the robustness of VAEs to adversarial attack. The first strand aims to tune the noise of the VAE latent space by up-weighting various regularisation terms (Willetts et al., 2020; Camuto et al., 2021b). The second aims to directly control the Lipschitz constants of the underlying networks (Barrett et al., 2021). In our framework we link these two strands: the Gaussian noise of a VAE's latent space affects the Lipschitz constants of the VAE's constituent networks.

6.2 Background

Variational Autoencoders: VAEs (Rezende et al., 2014a; Kingma and Welling, 2014), and the models they have inspired (Alemi et al., 2017; Kumar and Poole, 2020; Chen et al., 2018; Willetts et al., 2020), are deep latent variable models. Using $\mathbf{x} \in \mathcal{X}$ to denote data and $\mathbf{z} \in \mathcal{Z}$ to denote the latents with associated prior $p(\mathbf{z})$, a VAE simultaneously learns both a forward generative model, $p_\theta(\mathbf{x}|\mathbf{z})$, and an amortised approximate posterior distribution, $q_\phi(\mathbf{z}|\mathbf{x})$ (where θ and ϕ correspond to their respective parameters) which are typically implemented using neural networks [1]. These models are referred to as the decoder and encoder respectively, and a VAE can be thought of as a deep stochastic autoencoder. Under this autoencoder framework, one typically takes the reconstructions as deterministic, corresponding to the mean of the decoder, namely $g_\theta(\mathbf{z}) := \mathbb{E}_{p_\theta(\mathbf{x}|\mathbf{z})}[\mathbf{x}]$, a convention we adopt.

A VAE is trained by maximizing the evidence lower bound (ELBO) $\mathcal{L} = \mathbb{E}_{p_\mathcal{D}(\mathbf{x})}[\mathcal{L}(\mathbf{x})]$,

[1] Notation: We use bold letters to denote vectors or matrices and non-bolded letters to denote scalars. Matrices are capitalised.

where

$$\mathcal{L}(\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \tag{6.1}$$

and $p_{\mathcal{D}(\mathbf{x})}$ is the empirical data distribution. The optimisation is carried out using stochastic gradient ascent with Monte Carlo samples to evaluate expectations over $\mathbf{z}$, typically employing the reparameterisation trick (Kingma and Welling, 2014). For example, for a Gaussian $q_\phi(\mathbf{z}|\mathbf{x})$, we draw samples as $\mathbf{z} = \boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\epsilon} \circ \boldsymbol{\sigma}_\phi(\mathbf{x}), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, where $\circ$ is the element-wise product.

Gaussian Spaces: A *Gaussian Space*, denoted $L_2(\mathbb{R}^n, \gamma)$ is an L_2 space, the space of square-integrable functions $f : \mathbb{R}^n \to \mathbb{R}$, equipped with the Gaussian measure $\gamma(x) = \prod_i \mathcal{N}(x_i|0, 1)$. This space has an inner product between two functions f and g: $\langle f, g\rangle = \mathbb{E}_{\gamma(x)}\left[f(x)g(x)\right]$.

On the real line, *the Hermite polynomials* form an *orthogonal basis* for the space $L_2(\mathbb{R}, \gamma)$. These are the polynomials of degree k that satisfy (Janson, 1997):

$$\mathbb{E}_{\gamma(x)}[H_k(x)H_m(x)] = k!\mathbb{1}\{m = k\} \tag{6.2}$$

These polynomials can also be defined recursively, in a manner that allows a more intuitive understanding of their polynomial nature:

$$H_{k+1}(x) = xH_k(x) - kH_{k-1}(x) \tag{6.3}$$
$$H_0(x) = 1, H_1(x) = x$$

We have for instance

$$H_2(x) = x^2 - 1, H_3(x) = x^3 - 3x, H_4(x) = x^4 - 6x^2 + 3$$

A function $f \in L_2(\mathbb{R}, \gamma)$ in this space can be expressed as a weighted sum of these polynomial functions, where $\hat{f}(k)$ are the *Hermite coefficients*:

$$f = \sum_{k\in\mathbb{N}} \frac{1}{k!}\hat{f}(k)H_k, \qquad \hat{f}(k) = \langle f, h_k\rangle = \mathbb{E}_{\gamma(x)}\left[f(x)H_k(x)\right]. \tag{6.4}$$

6.3 Gaussian Spaces and VAEs

We want to bring the tools of Gaussian space analysis to bear on VAEs, by viewing the latent space of the VAE as being equipped with a Gaussian measure. The VAE encoder parameterises an amortised (per-datapoint) posterior Gaussian distribution with mean $\mu_\phi(\mathbf{x})$ and *diagonal variance* $\text{diag}(\sigma^2_\phi(\mathbf{x}))$. Thus we can view the latent space of a VAE as being equipped with a Gaussian measure that varies on a *per-datapoint basis*. This measure is a multivariate general Gaussian measure with mean $\mu_\phi(\mathbf{x})$ and *diagonal variance* $\text{diag}(\sigma^2_\phi(\mathbf{x}))$. The encoder in some sense 'indexes' over the range of possible Gaussian measures in a data-dependent manner. The decoder, which acts on this latent space, would then be a member of some Gaussian space (the measure for which depends on $\mathbf{x}$).

As we were unable to find a comprehensive resource for spaces equipped with non-standard Gaussian measures we must first derive results for Gaussian spaces equipped with a *general* Gaussian measure (with mean μ and variance σ^2) before we apply Gaussian space analysis to VAEs. We start with results for functions acting on a univariate space, $f : \mathbb{R} \to \mathbb{R}$.

Proposition 6.1. *If $x \sim \mathcal{N}(\mu, \sigma^2)$, then we have that $\hat{x} = (x - \mu)/\sigma \sim \mathcal{N}(0, 1)$ and the Hermite polynomials are the polynomials of degree k that satisfy:*

$$\mathbb{E}_{\gamma(\hat{x})}[H_k(\hat{x})H_m(\hat{x})] = k!\mathbb{1}\{m = k\},$$

where γ is the standard Gaussian measure. The family $\left\{\frac{1}{\sqrt{k!}}H_k(\hat{x}) : k \geq 0\right\}$ is then an orthonormal basis for $L_2(\mathbb{R}, \mathcal{N}(\mu, \sigma^2))$.

For the proof see Appendix D.1.1[2]. As such the polynomials $H_n(\hat{x})$ are the orthogonal polynomials for the $\mathcal{N}(\mu, \sigma^2)$ measure. We can calculate each of these polynomials by substituting x for $\hat{x} = (x - \mu)/\sigma$ in Equation (6.3). In higher dimensions ($\mathbb{R}^n$), for a function $f : \mathbb{R}^n \to \mathbb{R}$, we assume that we have a *diagonal* standard deviation $\mathbf{S}$, as this form of the standard deviation is most directly applicable to how VAEs are used in practice.

We denote the multivariate Gaussian measure as $\mathcal{N}(\boldsymbol{\mu}, \mathbf{S}^2)$ with covariance matrix

[2] All proofs are presented in Appendix D.1

populated by the element-wise square of $\mathbf{S}$. In this case the basis can be expressed using a multi-index $\alpha \in \mathbb{N}^n$ (the sets of size n of non-negative integers), where

$$|\alpha| = \sum_{i=1}^{d} \alpha_i, \qquad \mathbf{v}^\alpha = \prod_{i=1}^{n} v_i^{\alpha_i}, \qquad \alpha! = \prod_{i=1}^{n} \alpha_i!, \qquad g^{(\alpha)}(\mathbf{x}) = \frac{\partial^{|\alpha|} g}{\partial x_1^{\alpha_1} \dots \partial x_n^{\alpha_n}}.$$

The α^{th} multivariate Hermite polynomial (denoted $\mathcal{H}_\alpha$) for the measure $\mathcal{N}(\boldsymbol{\mu}, \mathbf{S}^2)$ can then be expressed as the product of univariate polynomials indexed by α:

$$\mathcal{H}_\alpha((\mathbf{x} - \boldsymbol{\mu})\mathbf{S}^{-1}) = \mathcal{H}_\alpha(\hat{\mathbf{x}}) = \prod_i H_{\alpha_i}((x_i - \mu_i)/\sigma_i),$$

This stems from the fact that each H_{α_i} forms a basis in the univariate case, and that we assume a diagonal covariance, meaning that we can obtain the basis for $\mathbb{R}^n$ simply by tensorisation $H_{\alpha_1} \otimes \cdots \otimes H_{\alpha_n}$. For the sake of completeness, in Appendix D.2 we derive the Hermite polynomials for a Gaussian space with a *full covariance matrix*.

Because the set of $\mathcal{H}_\alpha$ form an orthogonal basis, we can express functions f in the Gaussian space $L_2(\mathbb{R}^n, \mathcal{N}(\boldsymbol{\mu}, \mathbf{S}^2))$ as a weighted sum of these functions:

$$f = \sum_{\alpha \in \mathbb{N}^n} \frac{1}{\alpha!} \hat{f}(\alpha) \mathcal{H}_\alpha, \qquad \hat{f}(\alpha) = \mathbb{E}_{\gamma(\hat{\mathbf{x}})} \left[f(\mathbf{x}) \mathcal{H}_\alpha(\hat{\mathbf{x}}) \right], \qquad \gamma(\hat{\mathbf{x}}) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \mathbf{S}^2). \tag{6.5}$$

Using these results, we can express the Hermite cofficients for a general Gaussian space in terms of the underlying measure's standard deviation $\mathbf{S}$ and the derivatives of a function f in the space.

Proposition 6.2. *Assume we have a function f in Gaussian space with diagonal covariance, $f \in L_2(\mathbb{R}^n, \mathcal{N}(\boldsymbol{\mu}, \mathbf{S}^2))$. Further assume that f is in C^∞, the class of infinitely differentiable functions. For $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{S}^2)$, then we have that $\hat{\mathbf{x}} = (\mathbf{x} - \boldsymbol{\mu})\mathbf{S}^{-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and the Hermite coefficients can be expressed as:*

$$\hat{f}(\alpha) = (\text{diag}(\mathbf{S}))^\alpha \, \mathbb{E}_{\gamma(\hat{\mathbf{x}})} \left[f^{(\alpha)}(\mathbf{x}) \right], \gamma(\hat{\mathbf{x}}) = \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \mathbf{S}^2).$$

We can now express the variance of a function $L_2(\mathbb{R}^n, \mathcal{N}(\boldsymbol{\mu}, \mathbf{S}^2))$ as a sum of function derivatives, which in turns allows us to connect this variance to the Fourier domain.

Theorem 6.3. *Assume we have a function f in Gaussian space with diagonal covariance, $f \in L_2(\mathbb{R}^n, \mathcal{N}(\boldsymbol{\mu}, \mathbf{S}^2))$. Further assume that f is in C^∞, the class of infinitely differentiable functions and is L_2 integrable with respect to the Lebesgue measure. For $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{S}^2)$, then we have that $\hat{\mathbf{x}} = (\mathbf{x} - \boldsymbol{\mu})\mathbf{S}^{-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ such that $\gamma(\hat{\mathbf{x}}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{S}^2)$. The variance of f can be expressed as*

$$\mathrm{Var}(f) = \sum_{|\alpha| \geq 1} \frac{(\mathrm{diag}(\mathbf{S}))^{2\alpha}}{\alpha!} \left| \mathbb{E}_{\gamma(\hat{x})} \left[f^{(\alpha)}(\mathbf{x}) \right] \right|^2 = \sum_{|\alpha| \geq 1} \frac{(\mathrm{diag}(\mathbf{S}))^{2\alpha}}{\alpha!} \left| \int_{\mathbb{R}^n} (\boldsymbol{\omega})^\alpha \mathcal{F}(\boldsymbol{\omega}) \overline{\mathcal{P}(\boldsymbol{\omega})} d\boldsymbol{\omega} \right|^2$$

where $\overline{\mathcal{P}}$ is the complex conjugate of the Fourier transform of the Gaussian measure $\mathcal{N}(\boldsymbol{\mu}, \mathbf{S}^2)$ given by $\mathcal{P}(\boldsymbol{\omega}) = \det(\mathbf{S})\mathcal{G}(\boldsymbol{\omega}\mathbf{S})e^{-i\boldsymbol{\omega}\boldsymbol{\mu}\mathbf{S}^{-1}}$ (where $\mathcal{G}$ is the Fourier transform of the standard Gaussian measure γ) and $\mathcal{F}$ is the Fourier transform of f, and $\alpha \in \mathbb{N}^n$.

We also note that even if a function is *not* infinitely differentiable, we can also express $\mathrm{Var}(f)$ as a weighted sum of the Hermite coefficients (see the Proof of Theorem 6.3):

$$\mathrm{Var}(f) = \sum_{|\alpha| \geq 1} \frac{1}{\alpha!} \left| \hat{f}(\alpha) \right|^2. \tag{6.6}$$

If we now assume that f is infinitely differentiable, then we obtain the intuitive result that high-frequency functions correspond to larger Hermite coefficients associated with higher degree Hermite polynomials. This can be deduced by combining Proposition 6.2 and Theorem 6.3:

$$\left| \hat{f}(\alpha) \right|^2 = \left| \int_{\mathbb{R}^n} (\mathrm{diag}(\mathbf{S}))^\alpha (\boldsymbol{\omega})^\alpha \mathcal{F}(\boldsymbol{\omega}) \overline{\mathcal{P}(\boldsymbol{\omega})} d\boldsymbol{\omega} \right|^2. \tag{6.7}$$

As the variance of the underlying Gaussian measure increases, larger degree polynomials contribute more heavily to $\mathrm{Var}(f)$. This is captured in the terms $(\mathrm{diag}(\mathbf{S}))^{2\alpha}$ of Equation (6.7), meaning that increases in $|(\mathrm{diag}(\mathbf{S}))|$ disproportionately increase large α terms, i.e the coefficients associated with higher degree polynomial terms. More succinctly, Gaussian spaces with large variances naturally induce larger Hermite coefficients associated with higher degree polynomials, which themselves are associated with higher frequency components in the Fourier domain.

Lipschitzness Finally we redevelop Nash's Poincaré inequality for general Gaussian spaces to show that the variance of a function $f \in L_2(\mathbb{R}^n, \mathcal{N}(\boldsymbol{\mu}, \mathbf{S}^2))$, assuming that this function is Lipschitz continuous, can be upper-bounded by using its Lipschitz constant.

Proposition 6.4. *Let $f \in L_2(\mathbb{R}^n, \mathcal{N}(\boldsymbol{\mu}, \mathbf{S}^2))$ be a function that is Lipschitz continuous with Lipschitz constant L. Further assume that $f^{(|\alpha|=1)} \in L_2(\mathbb{R}^n, \mathcal{N}(\boldsymbol{\mu}, \mathbf{S}^2))$, we have:*

$$\mathrm{Var}(f) \leq L^2 \|\mathbf{S}\|_2^2.$$

This bound is relatively tight in that for a measure $\mathcal{N}(\mathbf{0}, \mathbf{I})$ and a function $f(\mathbf{x}) = \frac{1}{n}\sum x_i$ the bound becomes an equality: $L = n^{-1}$ and $\mathrm{Var}(f) = n^{-2}$. We can now use these results for a general Gaussian space to study VAE decoders.

6.3.1 Gaussian Spaces and VAE decoders

The amortised posterior distributions of a VAE are typically chosen as Gaussians, where, to allow for the backpropagation of gradients, samples are reparameterised as:

$$\mathbf{z} = \boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\sigma}_\phi(\mathbf{x}) \circ \boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{6.8}$$

This sample is then fed into the decoder to calculate the likelihood term of the ELBO (see Eq (6.1)). Implicit in this arrangement is that the VAE decoder operates on samples $\mathbf{z}$ in latent space $\mathcal{Z}$ equipped with a Gaussian measure, captured by the $\boldsymbol{\epsilon}$ term in Eq (6.8).

Assuming the function parameterised by the decoder g, for a given point $\mathbf{x}$, is L_2 integrable with respect to the general Gaussian measure, i.e $g \in L_2(\mathbb{R}^n, \mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{x}), \boldsymbol{\sigma}_\phi(\mathbf{x})))$ (this is reasonable as we are effectively training the decoder to be in this space) then we can directly apply the theory of Gaussian spaces to the VAE decoder. Assuming we have a Gaussian likelihood with a fixed standard deviation σ_θ across dimensions, we express the likelihood for a single point using a bias-variance decomposition, which gives the sum

of an error term captured by the bias, and a regularisation term captured by the variance:

$$\frac{1}{\sigma_\theta^2}\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[(g_\theta(\mathbf{z})-\mathbf{x})^2\right] = \frac{1}{\sigma_\theta^2}\left(\left(\text{Bias}_{q_\phi(\mathbf{z}|\mathbf{x})}\left(g_\theta(\mathbf{z})\right)\right)^2 + \text{Var}_{q_\phi(\mathbf{z}|\mathbf{x})}(g_\theta(\mathbf{z}))\right) \tag{6.9}$$

$$\text{Bias}_{q_\phi(\mathbf{z}|\mathbf{x})}\left(g_\theta(\mathbf{z})\right) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[g_\theta(\mathbf{z})\right] - \mathbf{x}$$

$$\text{Var}_{q_\phi(\mathbf{z}|\mathbf{x})}\left(g_\theta(\mathbf{z})\right) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\left(g_\theta(\mathbf{z}) - \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[g_\theta(\mathbf{z})\right]\right)^2\right].$$

We can directly apply Theorem 6.3 to the decoder function if we assume that the decoder function is infinitely differentiable. This holds for networks with the Sigmoid activation function for instance Hornik (1991). We can then swap out $\text{diag}(\mathbf{S})$ for the encoder standard deviation $\boldsymbol{\sigma}_\phi(\mathbf{x})$ and repeatedly apply the Theorem to each output $g_{\theta,i}(\mathbf{z}), i \in 1,\ldots,d$ of the decoder. For an output i of the decoder and $\alpha \in \mathbb{N}^n$, we have that:

$$\text{Var}_{q_\phi(\mathbf{z}|\mathbf{x})}(g_{\theta,i}) = \sum_{|\alpha|\geq 1}\frac{(\boldsymbol{\sigma}_\phi(\mathbf{x}))^{2\alpha}}{\alpha!}\left|g_{\theta,i}^{(\alpha)}(\mathbf{z})\right|^2 = \sum_{|\alpha|\geq 1}\frac{(\boldsymbol{\sigma}_\phi(\mathbf{x}))^{2\alpha}}{\alpha!}\left|\int_{\mathbb{R}^n}(\boldsymbol{\omega})^\alpha \mathcal{F}_{\theta,i}(\boldsymbol{\omega})\overline{\mathcal{P}_\phi(\boldsymbol{\omega})}d\boldsymbol{\omega}\right|^2.$$

$\mathcal{P}_\phi$ is the Fourier transform of the Gaussian measure $\mathcal{N}(\mu_\phi(\mathbf{x}), \sigma_\phi^2(\mathbf{x}))$ and $\mathcal{F}_{\theta,i}$ is the Fourier transform of $g_{\theta,i}$.

As $\boldsymbol{\sigma}_\phi(\mathbf{x})$ increases, particularly for $\boldsymbol{\sigma}_\phi(\mathbf{x}) \geq 1$, larger α terms in the sum, and large frequencies to the power α will begin to be disproportionately penalised in the Var term of the bias variance decomposition of the VAE likelihood. Thus larger $\boldsymbol{\sigma}_\phi(\mathbf{x})$ will result in a larger penalisation of the high-frequency components of the decoder function. We note that this penalisation is modulated on a *per-datapoint basis* by $\boldsymbol{\sigma}_\phi(\mathbf{x})$, but that VAEs which on average have larger encoder variances over training inputs will learn lower frequency decoders. Because of the link between the decoder variance, the Hermite polynomials, and the frequency domain (see Equation (6.7)), the lower frequency function learned by the decoder, for each $\mathbf{z}|\mathbf{x}$, *also* corresponds to a function that can be described as a lower degree polynomial.

To demonstrate these findings, in Figure 6.1 we use a fixed encoder variance σ_ϕ^2, uniform across dimensions, on VAEs, with Sigmoid activation function in their networks, trained on data from the $\text{sinc}(5t), t \in [-1,1]$ function. As σ_ϕ increases, we can quali-

tatively ascertain that the decoder function loses higher frequency components and that the optimal polynomial to describe the decoder function (approximated by polynomial regression) *decreases* in its degree. Quantitatively measuring the Fourier transform of the decoder here is simple, we collect the reconstruction of all inputs, ordering by increasing t. We then take the fast Fourier transform (FFT) of this aggregate reconstructed function. We show this in Figure 6.2 where as σ_ϕ increases, the decoder learns a lower frequency Fourier representation.

Though our theoretical results hold for classes of infinitely differentiable neural networks, we empirically confirm they still hold for more complex neural network architectures with ReLU activation functions, trained on large multivariate datasets. In Figure 6.3 we show that for a fixed encoder variance σ_ϕ^2, as σ_ϕ increases reconstructed CelebA images from a convolutional VAE become more similar and start to lose diversity – the images lose mid to high level frequencies as measured by a 2D-FFT – suggesting that the decoder learns a lower frequency function. To support this, we plot the mean 1D-Non-uniform discrete Fourier transform (NUDFT) (Bagchi and Mitra, 1999) across the d dimensions of the output, which shows that models trained with larger σ_ϕ learn functions that on average are of much lower frequency content than smaller σ_ϕ models. This demonstrates that the multi-dimensional output decreases in its high-frequency content over the d output dimensions, as predicted by our theory. In Figure D.4.2 of the Appendix we show that this also holds for fully-connected VAEs trained on multivariate sinusoids.

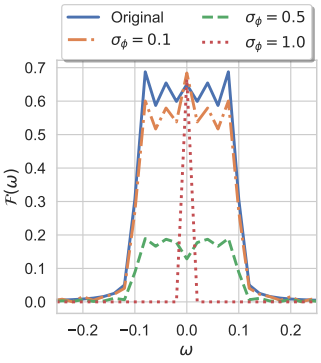
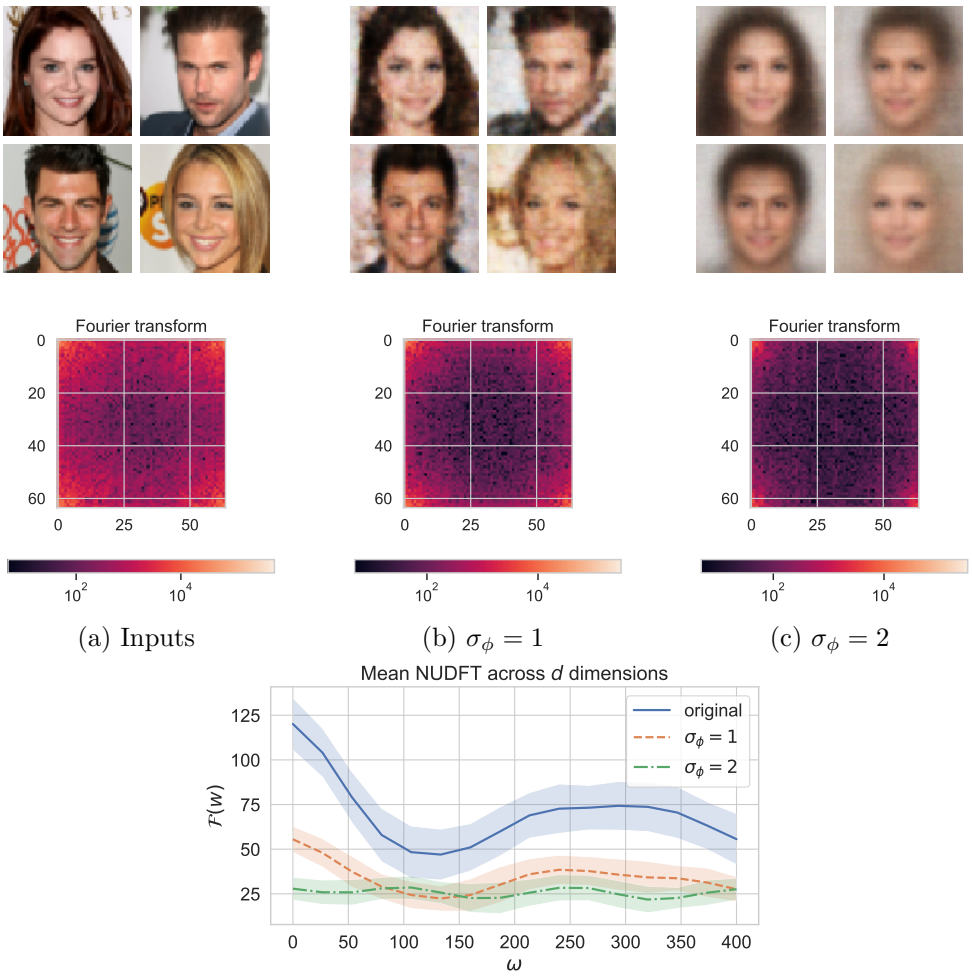


Figure 6.2: FFT of the VAE reconstructions in Figure 6.1. For larger $\boldsymbol{\sigma}_\phi$, the spectrum loses high frequencies.

6.3.2 Gaussian Spaces and VAE encoders

Here we use the analysis developed in previous section to analyze the Harmonic content of the VAE encoder. In the typical VAE setup, we don't usually consider the case of Gaussian noise on inputs and we make no assumptions as to the underlying measure of



(a) Inputs (b) $\sigma_\phi = 1$ (c) $\sigma_\phi = 2$

Figure 6.3: We train convolutional VAEs (see Appendix D.3 for network details), with ReLU activation functions in their networks, on the CelebA dataset, with a 64 dimension latent space $\mathcal{Z}$. As before we fix the encoder σ_ϕ. **[first + second rows]** We show 4 images and the 2D-FFT of the image in the upper right quadrant for a) the training images used b) VAE reconstructions of the 4 images when $\sigma_\phi = 1$ c) VAE reconstructions for $\sigma_\phi = 2$. As σ_ϕ increases mid-high level frequencies are increasingly dampened relative to the original image. **[bottom row]** Positive components of the mean 1D-NUDFT of the d dimensions of the output of these models (calculated across 2000 images for each of the $d = 12288$ dimensions). Shading corresponds to the std. dev. over d dimensions.

the input space; such that we cannot directly apply the previous Gaussian space analysis to the VAE encoder.

The perspective we've established, however, of the latent space as a Euclidean space equipped with a general Gaussian measure, and the perspective of the decoder as belonging to Gaussian space equipped with this same measure, can inform our analysis. The encoder of the VAE effectively parameterises the latent space measure for each input $\mathbf{x}$ such that the function the encoder is learning is the measure $\mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{x}), \boldsymbol{\sigma}_\phi(\mathbf{x}))$. In Theorem 6.3 we

have already established that the Fourier transform of this measure is given by

$$\mathcal{P}(\boldsymbol{\omega}) = \det(\boldsymbol{\Sigma}_\phi(\mathbf{x}))\mathcal{G}(\boldsymbol{\omega}\boldsymbol{\Sigma}_\phi(\mathbf{x}))e^{-i\boldsymbol{\omega}\boldsymbol{\mu}_\phi(\mathbf{x})\boldsymbol{\Sigma}_\phi^{-1}(\mathbf{x})},$$

where $\boldsymbol{\Sigma}_\phi(\mathbf{x})$ is a diagonal matrix with the elements of $\boldsymbol{\sigma}_\phi(\mathbf{x})$ on its diagonal. Much like the decoder, as the encoder variance decreases, the function parameterised by the encoder increases in amplitude in its high-frequency components. However, here the frequency content of the encoder function is *intrinsic* to the Gaussian measure, whereas the frequency content of the decoder is a result of the relative weighting of the decoder variance in the VAE likelihood, and as such is enforced by optimisation. Note that $e^{-i\boldsymbol{\omega}\boldsymbol{\mu}_\phi(\mathbf{x})\boldsymbol{\Sigma}_\phi^{-1}(\mathbf{x})}$ only modulates the phase of the Fourier transform and does not alter the amplitude of its frequency content. As such the mean does not affect the frequency spectrum of the posterior Gaussian. In Figure D.4.1 of the Appendix we demonstrate that for a univariate Gaussian, decreasing the variance increases the magnitude of high-frequency components in the Fourier domain, whereas altering the mean has no such effect.

This perspective gives us an idea of how to alter the harmonic content of the measure parameterised by the encoder *overall*. If instead we want to modulate the frequency content of the *individual networks* that are used to parameterise the Gaussian measure we need a different perspective. In the next section we show that using the theory developed previously, adding Gaussian noise to the input data offers a simple and effective method for modulating the frequency content of the encoder mean.

Noisy inputs Let us now assume that we add Gaussian noise to the data $\mathbf{x}$ such that our input for each point $\mathbf{x}$ is a Gaussian space with a Gaussian measure with mean $\mathbf{x}$, variance σ^2:

$$\tilde{\mathbf{x}} = \mathbf{x} + \sigma\boldsymbol{\nu}, \boldsymbol{\nu} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

Under this noisy input, the expectation of the KL between the amortised posterior distribution and the unit Gaussian prior can be written as:

$$\mathbb{E}_{\gamma(\boldsymbol{\nu})}\left[\mathrm{KL}(q_\phi(\mathbf{z}|\tilde{\mathbf{x}})\|p(\mathbf{z}))\right] = \frac{1}{2}\,\mathbb{E}_{\gamma(\boldsymbol{\nu})}\left[\log\left(\frac{1}{\prod_{i=1}^n \sigma_{\phi,i}(\tilde{\mathbf{x}})}\right) - n + \sum_{i=1}^n \sigma_{\phi,i}(\tilde{\mathbf{x}}) + \left\|\boldsymbol{\mu}_\phi(\tilde{\mathbf{x}})\right\|_2^2\right].$$

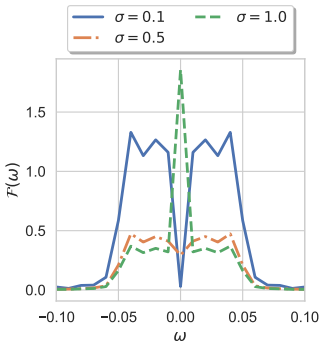


Figure 6.4: FFT of μ_ϕ of VAEs trained on $\text{sinc}(5t) + \mathcal{N}(0, \sigma^2\mathbf{I})$. For larger σ the spectrum loses high frequency content.

We now have a Gaussian space variance $\text{Var}(\boldsymbol{\mu}_\phi) = \mathbb{E}_{\gamma(\boldsymbol{\nu})}\left[\left\|\boldsymbol{\mu}_\phi(\tilde{\mathbf{x}})\right\|_2^2\right]$ penalised at each iteration. As before for the decoder we can directly apply Theorem 6.3 to the encoder mean function, by swapping out $\text{diag}(\mathbf{S})$ for the data standard deviation $\sigma\mathbf{I}$ and repeatedly applying the Theorem to each output $\mu_i(\tilde{\mathbf{x}}), i \in 1, \ldots, n$ of the encoder. We can expect two things to happen as σ increases. The frequency content of the encoder mean should decrease, and generally the function for each $\mathbf{x}$ can be described using an increasingly lower degree polynomial. Thus we can add Gaussian noise to a VAE's inputs to modulate the frequency content of the encoder mean function. We demonstrate these findings in Figure 6.4. Note that though our theory is not directly applicable to the encoder variance network, if we fix σ_ϕ so as to modulate the variance of the decoder, then we have full control over the variance of the encoder by adding noise to the VAE inputs.

6.3.3 The KL and frequency content

In previous experiments we fixed the variance of the encoder for analytical purposes. In practice however we often want to learn this variance, and as such, need tools by which we can modify the frequency content of the encoder and decoder whilst learning the encoder variance. As we now show, the frequency content of both the encoder mean and decoder networks can be controlled by altering the weighting of KL in the ELBO.

Consider the β-VAE setting (Higgins et al., 2017b), whereby the KL component of the ELBO is pre-multiplied by a factor β. The strength of this penalisation changes the optimal encoder variance $\sigma_\phi(x)$: as β increases we can expect $\sigma_\phi(x)$ to tend to 1 – see Theorem 2 of (Camuto et al., 2020) for a formal statement of this. This then means that we can modulate the scale of the latent noise experienced by the model during training, where $\sigma_\phi(x)$ is learnt and no longer fixed as in previous experiments. Because increasing β tends to increase $\sigma_\phi(x)$, our theory would predict that gradual increases in β would be accompanied by the decoder losing higher-frequency content in the Fourier domain.

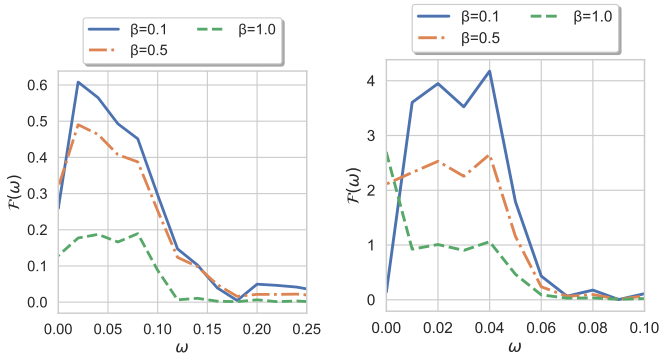


Figure 6.5: Positive components of Figure 6.4 with variable β penalisation on the KL term of the ELBO. As β increases higher frequencies are increasingly dampened for the decoder [left] and encoder [right]

We also note that the variance of the encoder mean network $\text{Var}(\boldsymbol{\mu}_\phi) = \mathbb{E}_{\gamma(\boldsymbol{\nu})}\left[\left\|\boldsymbol{\mu}_\phi(\tilde{\mathbf{x}})\right\|_2^2\right]$ experiences a greater penalisation as the weight β increases, as seen in the decomposition of the KL in section 6.3.2. This means that we expect larger KL weightings to induce encoder mean functions with lower frequency content (when the input space is a Gauss space). See Figure 6.5 for a demonstration of both these claims.

6.3.4 VAE Lipschitzness

We know that by Proposition 6.4 the variance of a function (that is once differentiable) provides a relatively tight lower bound on the Lipschitz constant of a neural network. As the encoder variance $\boldsymbol{\sigma}_\phi^2(\mathbf{x})$ increases, the variance of the decoder function $\text{Var}_{q_\phi(\mathbf{z}|\mathbf{x})}\left(g_\theta(\mathbf{z})\right)$ decreases, which by Proposition 6.4 will lead to a smaller lower bound on the Lipschitz constant of the decoder. Due to the computational costs of estimating the Lipschitz constants of networks, we restrict our empirical analysis here to fully-connected VAEs and use *layer-wise* LipSDP (Fazlyab et al., 2019) to obtain estimates. In Appendix D.4.3 we confirm that regulating the encoder variance allows for us to impose a soft constraint of the Lipschitzness of the VAE decoder. We also show that adding Gaussian noise on data, gives us finer control on the Lipschitz constant of the encoder. Finally, we also demonstrate that this modulation of the encoder and decoder Lipschitzness is a purveyor of adversarial robustness for VAEs.

6.3.5 Potential Limitations

Our results on Lipschitzness would be stronger if Proposition 6.4 offered an upper bound on the Lipschitz constant, in that we could certify a maximum Lipschitz constant. Also, the encoder variance does not *directly* affect the frequency content of the encoder, only secondarily. Noise injections on data are needed to provide full control over the frequency content and Lipschitzness of the VAE networks, whereas a single parameter to control would have been a stronger result. We also recognise that the results in Figure D.4.3 would have been stronger if the variance of the likelihood degradation results was such that the shading of the curves did not overlap for the different values of σ and σ_ϕ. Our study does not have a direct societal impact as it is largely theoretical.

6.4 Conclusion

Using the lens of Gaussian spaces, we demonstrated that the variance of the latent encodings features as an important parameter for regularising VAE decoders, controlling their Lipschitz constants, and improving their adversarial robustness. By applying the same framework to the VAE encoder we show that simply adding Gaussian noise to VAE inputs offers the same control over the VAE encoder. This work lays the foundation for analysing the effect of Gaussian priors on VAEs and offers a novel framework from which to understand the functions learned by a VAE's networks. It also paves the way for the study of the effect of *non-Gaussian* priors on VAE networks.

Acknowledgements

We thank Professor Svante Janson, Professor Wilfreda Urbina-Romero, and Professor Tom Alberts for their invaluable guidance on Gaussian Spaces. This research was directly funded by the Alan Turing Institute under Engineering and Physical Sciences Research Council (EPSRC) grant EP/N510129/1. Alexander Camuto was supported by an EPSRC Studentship.

CHAPTER 7

CONCLUSION

This thesis has presented a comprehensive study of the effects of Gaussian Noise Injections on neural networks, highlighting their regularising effects (§3) their conference of adversarial robustness (§5) as well as the subtle and unexpected bias they introduce into gradient descent related optimisation algorithms (§4). Our study ends with the presentation of Gaussian spaces as an elegant and simple tool for the study of the effect of such noise injections (§6).

In particular, §3, *The Explicit Regularisation of Gaussian Noise Injections* demonstrates that the marginal of Gaussian noise injections (GNIs) corresponds to an added penalisation that regulates the frequency content of the function learned by neural networks. This work uses first and second order approximations to approximate this marginal, and demonstrates empirically that first and second effects dominate other higher-order effects of GNIs for a range of neural architectures and activations functions. Though this approximation is inelegant, and actually quite involved to derive (see the associated proof) it lays the foundation for the final presented work, which offers more abstract mathematical tooling that more simply connects (GNIs) to the frequency domain, without making such approximations.

Indeed one can view §6, *Variational Autoencoders: a Harmonic Perspective* as a work that more elegantly and succinctly presents the ideas of *The Explicit Regularisation of Gaussian Noise Injections* using the theory surrounding Gaussian spaces; which are mea-

sure spaces equipped with a Gaussian measure. Though the work focuses on Variational Autoencoders, and the way Gaussian samples in the latent space then regulate the frequency content of encoder and decoder networks, the analysis can actually be used to study the supervised learning setting of the first work. If we view data experiencing noise injections as a form of Gaussian Space, with mean centered on the data point and variance corresponding to the variance of the noise injection, then we can use the tools of harmonic analysis to re-derive some of the findings of the Chapter 3.

§5, studies the effects of GNIs that are experienced at test-time, post-training. The work, in conjunction with Willetts et al. (2019), presents the first study of the properties of VAEs that contribute to adversarial robustness. By viewing the sampling that takes place in the latent space as a form of Gaussian noise injection on the VAE decoder, we build on the work of Cohen et al. (2019) to introduce the concept of r-robustness. A VAE is r-robust to a perturbation δ at a point x if the VAE's reconstruction under the distorted input will fall within an r-ball (under a chosen norm) of the undistorted reconstruction more than half the time. We then derive a margin on δ_x such that for all distortion with $\|\delta_x\|$ less than the margin a VAE is r-robust. The form of this margin depends on the minimum of the posterior's standard deviation and on the reciprocal of the Frobenius norm of the Jacobian of the mean of the posterior. Using this margin we can now understand formally how a simple method for robustness in VAEs, β-VAEs (which simply multiply the KL of the ELBO by a pre-factor β), obtain increased robustness. β-penalisation both naturally increases the minimum of the posterior standard deviation, and decreases the Frobenius norm of the Jacobian of the mean.

Variational Autoencoders: a Harmonic Perspective also expands on this work, using the connection between GNIs, the frequency domain, the Lipschitz-constant of a function experiencing GNIs, and the work of Barrett et al. (2021) to more simply derive this same result. By the harmonic analysis we present, VAEs with larger variance posteriors (i.e larger variance latent samples) – and by extension β-VAEs with larger β – tend to learn lower frequency decoder and encoder functions. Our re-tooling of Nash's Poincaré inequality then allows us to show that these lower frequency functions also have correspondingly lower Lipschitz constants. Barrett et al. (2021) demonstrate formally that the r-robustness

margin of VAEs decreases for smaller Lipschitz encoder and decoder functions. As such this final work also presents a link between Chapters 3 and 5 by demonstrating that the frequency modulation conferred by GNIs also improves the adversarial robustness of deep probabilistic models.

Finally §4, *Asymmetric Heavy Tails and Implicit Bias in Gaussian Noise Injections*, nuances the benefits conferred by GNIs by showing that the implicit effect, which is the remainder of the marginal studied in *The Explicit Regularisation of Gaussian Noise Injections*, introduces bias in Stochastic Gradient Descent (SGD) which degrades neural network performance. Indeed, rather counter-intuitively, though the forward pass of neural network experiences Gaussian noise during GNIs, the backward pass (the gradients) can experience asymmetric and heavy tailed noise. By modelling the combined gradient noise from mini-batching *and* GNIs as being drawn from a skewed α-stable distribution, a type of heavy-tailed distribution with infinite moments above a degree α (this approximation is reasonable as Şimşekli et al. (2019b) demonstrate that the gradient noise stemming from mini-batching is α-stable), we derive results that show that neural networks trained with GNIs experience bias that can lead them away from minimisers of the objective function they are optimising for. Thus the usage of GNIs introduces a trade-off between bias and regularisation, which practitioners should bear in mind when using such methods. Note that these results are also translatable to the deep probabilistic model setting. The analysis conducted is directly applicable to the VAE decoder setting, which effectively experience GNIs due to the sampling that takes place in the VAE latent space during training. The gradient noise on these networks will also be heavy-tailed and skewed and likely to introduce bias during training. Mitigating this bias in the VAE setting is a potentially useful extension of this work.

7.1 Extensions and Applications

Though we studied Gaussian noise principally for analytic simplicity (Gaussian distributions have finite moments and are amenable to Taylor-expansion type analysis which form the crux of our first two works), the methodologies and ideas presented lay the foundation for the study of other forms of noise in neural networks. In particular by framing

noise injections as a form of *measure space* we can extend some of these ideas to other well-behaved distributions, such as the Bernoulli distribution (used in *dropout*). By then viewing noised neural network hidden layers as mappings between measure spaces, we can use the analysis of our final work to simplify the analysis of our first and second works, without resorting to first and second order approximations.

The connection between Sobolev spaces and GNIs detailed in §3 also potentially paves the way for studying neural networks experiencing GNIs from the perspective of Reproducing Kernel Hilbert Spaces (RKHS). Assuming certain regularity conditions, notably that the degree of the Sobolev space a neural network lies in is strictly greater than $d/2$ where d is the dimensionality of the input space, then the neural network will have an associated reproducing kernel that has been defined in closed form by Novak et al. (2018). The penalisation of the Sobolev norm, added to the loss function by GNIs, could lead to the definition of an 'optimal' function under GNI injections in terms of the reproducing kernel, as described by (Wahba, 1990).

In §5 the margin of robustness is defined as an L_2 norm within the input space. Though this margin is predictive of robustness for natural images, images themselves rarely take continuous values. Their pixel values lie in $\{0, 1\}$ for black and white binary images, and in the range of integers from 0 to 255 for images encoded using RGB. A more appropriate margin for such data might be in terms of the L_0 or 'Hamming' distance, representing a discrete step-wise pixel 'flip' from one integer to another. This is particularly relevant given the constructive proof of the existence of small-Hamming distance adversarial examples given by (Shamir et al., 2019).

7.2 Implications

This thesis already presents the ways in which the analysis of the effects of noise can be applied to deep probabilistic models, such as VAEs. By taking a more 'mechanical' and functional perspective of VAEs (rather than the traditional probabilistic perspective) we can derive interesting insights into the characteristics of the learnt encoding and decoding functions; in particular their robustness to adversaries, their frequency content, but also

the biases they experience when trained with SGD (as §4 is also applicable to the decoders of VAEs which experience GNIs on their 'inputs'). Further the choice of Gaussian posterior induces an inadvertent regularisation of VAE networks which as of yet had not been studied before §6. This suggests that the choice of prior and posterior in deep probabilistic models not only impacts the flexibility and suitability of the resulting probabilistic model to a given task, they also can strongly affect the neural networks that parameterise the distributions of the model. As a practitioner the choice of distribution prior and posterior is now two-fold, the first relates to their suitability to the task at hand, the second relates to their regularising effect (or conference of adversarial robustness) on the underlying networks.

In conversations with engineers at Mythic AI[1] we discovered that these works were being used to study neural networks trained on their proprietary hardware. The company is developing chips to accelerate neural network training which use some analogue components. The conversion between analogue and digital signals inadvertently introduces Gaussian noise during neural network training, the effects of which they are attempting to understand. In particular they are looking for ways in which they can limit the bias on SGD GNIs introduce. As such a natural extension of this thesis would be to develop methods which reduce the implicit bias of GNIs. Their observations also suggest something deeper, which is that the choice of hardware can unwittingly regularise (or introduce bias) to neural networks during training. This is particularly true for GPUs where race-conditions between threads mean that certain operations are non-deterministic and experience noise relative to CPU-based operations [2]. Given the prevalence of central limit theorem effects, our study of the effects of Gaussian noise could be applicable to the study of the effects of different hardware choices and drivers.

Finally the simplicity involved in implementing GNIs, and GNIs' low computational cost, means GNIs offer an attractive method for any practitioner to use as a means of limiting over-fitting and improving the adversarial robustness of their models.

[1] private correspondence

[2] The set of non-deterministic operations is currently rapidly changing and for the most up-to-date information the reader is best served reading the latest documentation for their respective GPU driver and favourite machine learning library.

APPENDIX A

APPENDIX: THE EXPLICIT REGULARISATION IN GAUSSIAN NOISE INJECTIONS

A.1 Technical Proofs . . . 108
A.1.1 Proof of Proposition 3.1 . . . 108
A.1.2 Proof of Theorem 3.2 . . . 108
A.1.3 Proof of Theorem 3.3 . . . 112
A.1.4 Regularisation in Regression Models and Autoencoders . . . 113
A.1.5 Regularisation in Classifiers . . . 113
A.2 Tikhonov Regularisation . . . 116
A.3 Measuring Calibration . . . 116
A.4 Classification Margins . . . 117
A.5 Model Capacity . . . 119
A.6 Additional Results . . . 120
A.7 Network Hyperparameters . . . 124

A.1 Technical Proofs

A.1.1 Proof of Proposition 3.1

Proof of Proposition 3.1. Recall that $\mathbf{h}$ denotes the vanilla activations of the network, those we obtain with no noise injection. Let us *not* inject noise in the final, predictive, layer of our network such that the noise on this layer is accumulated from the noising of previous layers.

We denote $\boldsymbol{\mathcal{E}}_k$ the noise accumulated at layer k from GNIs in previous layers, and potential GNIs at layer k itself. We denote $\mathcal{E}_{L,i}$ the i^{th} element of the noise at layer L, the layer to which we do not add noise. This can be defined as a Taylor expansion around the accumulated noise at the previous layer $L-1$:

$$\mathcal{E}_{L,i} = \sum_{|\alpha_L|=1}^{\infty} \frac{1}{\alpha_L!} \left(D^{\alpha_L} h_{L,i}(\mathbf{h}_{L-1}(\mathbf{x}))\right) \boldsymbol{\mathcal{E}}_{L-1}^{\alpha_L} \tag{A.1}$$

where we use α_L as a multi-index over derivatives.

Generally if we noise all layers up to the penultimate layer of index $L-1$ we can define the accumulated noise at layer k, $\boldsymbol{\mathcal{E}}_k$ recursively because Gaussian have finite moments:

$$\mathcal{E}_{k,i} = \epsilon_{k,i} + \sum_{|\alpha_k|=1}^{\infty} \frac{1}{\alpha_k!} \left(D^{\alpha_k} h_{k,i}(\mathbf{h}_{k-1}(\mathbf{x}))\right) \boldsymbol{\mathcal{E}}_{k-1}^{\alpha_k},\ i = 1, \dots, d_k,\ k = 0 \dots L-1 \tag{A.2}$$

where $\boldsymbol{\mathcal{E}}_0 = \boldsymbol{\epsilon}_0$ is the base case.

■

A.1.2 Proof of Theorem 3.2

Proof of Theorem 3.2. Let us first consider the Taylor series expansion of the loss function with the accumulated noise defined in Proposition 3.1. Denoting $\boldsymbol{\epsilon} = [\boldsymbol{\epsilon}_{L-1}, \dots, \boldsymbol{\epsilon}_0]$ we have:

$$\mathbb{E}_{\boldsymbol{\epsilon}} \left[\mathcal{L}(\mathbf{h}_L(\mathbf{x}) + \boldsymbol{\mathcal{E}}_L, \mathbf{y})\right] = \mathcal{L}(\mathbf{x}, \mathbf{y}) + \mathbb{E}_{\boldsymbol{\epsilon}} \left[\sum_{|\alpha|=1}^{\infty} \frac{1}{\alpha!} \left(D^{\alpha} \mathcal{L}(\mathbf{h}_L(\mathbf{x}), \mathbf{y})\right) \boldsymbol{\mathcal{E}}_L^{\alpha}\right] \tag{A.3}$$

Note that the dot product with the i^{th} element of the final layer noise $\mathcal{E}_{L,i}$ can be written as

$$\mathbb{E}_{\boldsymbol{\epsilon}}\left[\sum_{|\alpha|=1}^{\infty}\frac{1}{\alpha!}\left(D^{\alpha}\mathcal{L}(\mathbf{h}_L(\mathbf{x}),\mathbf{y})\right)\mathcal{E}_{L,i}\right]$$
$$=\mathbb{E}_{\boldsymbol{\epsilon}}\left[\sum_{|\alpha|=1}^{\infty}\frac{1}{\alpha!}\left(D^{\alpha}\mathcal{L}(\mathbf{h}_L(\mathbf{x}),\mathbf{y})\right)\left(\sum_{|\alpha_L|=1}^{\infty}\frac{1}{\alpha_L!}\left(D^{\alpha_L}h_{L,i}(\mathbf{h}_{L-1}(\mathbf{x}))\right)\boldsymbol{\mathcal{E}}_{L-1}^{\alpha_L}\right)\right] \tag{A.4}$$
$$=\mathbb{E}_{\boldsymbol{\epsilon}}\left[\sum_{|\alpha|=1}^{\infty}\frac{1}{\alpha!}\left(D^{\alpha}\mathcal{L}(\mathbf{h}_L(\mathbf{x}),\mathbf{y})\right)\left(\sum_{|\alpha_L|=1}^{\infty}\frac{1}{\alpha_L!}\left(D^{\alpha_L}h_{L,i}(\mathbf{h}_{L-1}(\mathbf{x}))\right)(\boldsymbol{\epsilon}_{L-1}+\dots)^{\alpha_L}\right)\right] \tag{A.5}$$

where the dots here denote the accumulated noise term on layer $L-1$ *before* we add the Gaussian noise $\boldsymbol{\epsilon}_{L-1}$. When looking at all elements of $\boldsymbol{\mathcal{E}}_L^{\alpha}$, not just the i^{th} element, note that this is essentially the Taylor series expansion of $\mathcal{L}$ around the series expansion of $\mathbf{h}_L$ around $\boldsymbol{\epsilon}_{L-1}$. We know that the product of the Taylor series of a composed function $f\circ g$ with the Taylor series of g is simply the Taylor series of f around $\mathbf{x}$ (Constantine and Savits, 1996). This can be deduced from the slightly opaque Faà di Bruno's formula, which states that for multivariate derivatives of a composition of functions $f:\mathbb{R}^m\to\mathbb{R}$ and $g:\mathbb{R}^d\to\mathbb{R}^m$ and a multi-index α (Constantine and Savits, 1996)

$$D^{\alpha}(f\circ g)(\mathbf{x})=\sum_{1\leq|\lambda|\leq|\alpha|}D^{\lambda}f(g(\mathbf{x}))\sum_{s=1}^{|\alpha|}\sum_{p_s(\lambda,\alpha)}(\alpha!)\prod_{j=1}^{s}\frac{(D^{l_j}g(\mathbf{x}))^{k_j}}{(k_j!)[l_j!]^{|k_j|}},$$

where $p_s(\lambda,\alpha)=\{(k_1,\dots,k_s);(l_1,\dots,l_s):|k_i|>0,\ 0\prec l_1\cdots\prec l_s,\ \sum_{i=1}^{s}k_i=\lambda\sum_{i=1}^{s}|k_i|l_i=\alpha\}$, where $\prec$ denotes a partial order.

Applying this recursively to each layer k, we obtain that,

$$\mathbb{E}_{\boldsymbol{\epsilon}}\left[\sum_{|\alpha|=1}^{\infty}\frac{1}{\alpha!}\left(D^{\alpha}\mathcal{L}(\mathbf{h}_L(\mathbf{x}),\mathbf{y})\right)\boldsymbol{\mathcal{E}}_L^{\alpha}\right]$$
$$=\mathbb{E}_{\boldsymbol{\epsilon}}\left[\sum_{k=0}^{L-1}\left[\sum_{|\alpha|=1}^{\infty}\frac{1}{\alpha_k!}\left(D^{\alpha_k}\mathcal{L}(\mathbf{h}_k(\mathbf{x}),\mathbf{y})\right)\boldsymbol{\epsilon}_k^{\alpha_k}\right]+\mathcal{C}((\mathbf{x},\mathbf{y});\boldsymbol{\epsilon})\right] \tag{A.6}$$

Here $\mathcal{C}(\boldsymbol{\epsilon},\mathbf{x},\mathbf{y})$ represents cross-interactions between the noise at each layer k $\boldsymbol{\epsilon}_k$ and the noise injections at preceding layers with index less than k. We can further simplify

the added term to the loss,

$$\begin{aligned}
&\mathbb{E}_{\boldsymbol{\epsilon}}\left[\sum_{k=0}^{L-1}\left[\sum_{|\alpha|=1}^{\infty}\frac{1}{\alpha_k!}\left(D^{\alpha_k}\mathcal{L}(\mathbf{h}_k(\mathbf{x}),\mathbf{y})\right)\boldsymbol{\epsilon}_k^{\alpha_k}\right]+\mathcal{C}((\mathbf{x},\mathbf{y});\boldsymbol{\epsilon})\right]\\
&=\sum_{k=0}^{L-1}\left[\sum_{|\alpha_k|=1}^{\infty}\frac{1}{2\alpha_k!}\left(D^{2\alpha_k}\mathcal{L}(\mathbf{h}_k(\mathbf{x}),\mathbf{y})\right)\mathbb{E}_{\boldsymbol{\epsilon}}\left[\boldsymbol{\epsilon}_k^{2\alpha_k}\right]\right]+\mathbb{E}_{\boldsymbol{\epsilon}}\left[\mathcal{C}((\mathbf{x},\mathbf{y});\boldsymbol{\epsilon})\right] \qquad \text{(A.7)}
\end{aligned}$$

The second equality comes from the fact that odd-numbered moments of $\boldsymbol{\epsilon}_k$, will be 0 and that $\mathcal{L}(\mathbf{x},\mathbf{y})=\mathcal{L}(\mathbf{h}_L(\mathbf{x}),\mathbf{y})$. The final equality comes from the moments of a mean 0 Gaussian, where j takes the values of the multi-index. Note that $\left[\boldsymbol{\epsilon}_k^{2\alpha_k}\right]$ are the even numbered moments of a zero mean Gaussian,

$$\mathbb{E}\left[\boldsymbol{\epsilon}_k^{2\alpha_k}\right]=[\sigma_k^{2\alpha_{k,1}}(2\alpha_{k,1}-1)!,\ldots,\sigma_k^{2\alpha_{k,d_k}}(2\alpha_{k,d_k}-1)!]^\intercal$$

Though these equalities can already offer insight into the regularising mechanisms of GNIs, they are not easy to work with and will often be computationally intractable. We focus on the first set of terms here where each $|\alpha_k|=1$, which we denote $R(\mathbf{x},\mathbf{w})$

$$\begin{aligned}
R(\mathbf{x},\mathbf{w})&=\sum_{k=0}^{L-1}\left[\sum_{|\alpha_k|=1}\frac{1}{2\alpha_k!}\left(D^{2\alpha_k}\mathcal{L}(\mathbf{h}_k(\mathbf{x}),\mathbf{y})\right)\mathbb{E}_{\boldsymbol{\epsilon}}\left[\boldsymbol{\epsilon}_k^{2\alpha_k}\right]\right]\\
&\approx\sum_{k=0}^{L-1}\left[\sum_{|\alpha_k|=1}\frac{\sigma_k^2}{2}\left(D^{2\alpha_k}\mathcal{L}(\mathbf{h}_L(\mathbf{x}),\mathbf{y})\right)\mathbf{J}_k^{2\alpha_k}(\mathbf{x})\right] \qquad \text{(A.8)}
\end{aligned}$$

The last approximation corresponds to the Gauss-Newton approximation of second-order derivatives of composed functions where we've discarded the second set of terms of the form

$$D\mathcal{L}(\mathbf{h}_L)(\mathbf{x})\left(D^2\mathbf{h}_L(\mathbf{h}_k(\mathbf{x}))\right)$$

. We will include these terms in our remainder term $\mathcal{C}$. For compactness of notation, we denote each layer's Jacobian as $\mathbf{J}_k\in\mathbb{R}^{d_L\times d_k}$. Each entry of $\mathbf{J}_k$ is a partial derivative of

$f^{\theta}_{k,i}$, the function from layer k to the i^{th} network output, $i = 1...d_L$.

$$\mathbf{J}_k(\mathbf{x}) = \begin{bmatrix} \frac{f^{\theta}_{k,1}}{\partial h_{k,1}} & \frac{f^{\theta}_{k,1}}{\partial h_{k,2}} & \cdots \\ \vdots & \ddots & \\ \frac{f^{\theta}_{k,d_L}}{\partial h_{k,1}} & & \frac{f^{\theta}_{k,d_L}}{\partial h_{k,d_k}} \end{bmatrix},$$

Again, for simplicity of notation $\mathbf{J}^{\alpha_k}_k$ selects the column indexed by $|\alpha_k| = 1$. Also note that the sum over $|\alpha_k| = 1$ effectively indexes over the diagonal of the Hessian of the Loss with respect to the L^{th} layer activations. We denote this Hessian as $\mathbf{H}_L(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{d_L \times d_L}$.

$$\mathbf{H}_L(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \frac{\partial^2 \mathcal{L}}{\partial h^2_{L,1}} & \frac{\partial^2 \mathcal{L}}{\partial h_{L,1} \partial h_{L,2}} & \cdots \\ \vdots & \ddots & \\ \frac{\partial^2 \mathcal{L}}{\partial h_{L,d_L} \partial h_{L,1}} & & \frac{\partial^2 \mathcal{L}}{\partial h^2_{L,d_L}} \end{bmatrix}$$

This gives us that

$$R(\mathbf{x}; \mathbf{w}) = \frac{1}{2} \sum_{k=0}^{L-1} \left[\sigma_k^2 \text{Tr} \left(\mathbf{J}_k^{\intercal}(\mathbf{x}) \mathbf{H}_L(\mathbf{x}, \mathbf{y}) \mathbf{J}_k(\mathbf{x}) \right) \right]$$

For notational simplicity we include the terms that R does not capture into the remainder $\mathbb{E}_{\boldsymbol{\epsilon}} \left[\mathcal{C}((\mathbf{x}, \mathbf{y}); \boldsymbol{\epsilon}) \right]$. We take expectations over the batch and have:

$$\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{B}} \left[\mathbb{E}_{\boldsymbol{\epsilon}} \left[\mathcal{L}(\mathbf{h}_L(\mathbf{x}) + \boldsymbol{\mathcal{E}}_L, \mathbf{y}) \right] \right] = \mathcal{L}(\mathcal{B}; \mathbf{w}) + R(\mathcal{B}; \mathbf{w}) + \mathbb{E}_{\boldsymbol{\epsilon}} \left[\mathcal{C}(\mathcal{B}; \boldsymbol{\epsilon}) \right] \tag{A.9}$$

$$R(\mathcal{B}; \mathbf{w}) = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{B}} \left[\frac{1}{2} \sum_{k=0}^{L-1} \left[\sigma_k^2 \text{Tr} \left(\mathbf{J}_k^{\intercal}(\mathbf{x}) \mathbf{H}_L(\mathbf{x}, \mathbf{y}) \mathbf{J}_k(\mathbf{x}) \right) \right] \right] \tag{A.10}$$

$$\mathbb{E}_{\boldsymbol{\epsilon}} \left[\mathcal{C}(\mathcal{B}; \boldsymbol{\epsilon}) \right] = \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{B}} \left[\mathbb{E}_{\boldsymbol{\epsilon}} \left[\sum_{|\alpha|=1}^{\infty} \frac{1}{\alpha!} \left(D^{\alpha} \mathcal{L}(\mathbf{h}_L(\mathbf{x}), \mathbf{y}) \right) \boldsymbol{\mathcal{E}}_L^{\alpha} \right] \right] - R(\mathcal{B}; \mathbf{w}) \tag{A.11}$$

This concludes the proof. ■

A.1.3 Proof of Theorem 3.3

Proof of Theorem 3.3. Because $f \in W_{\mu}^{1,2}(\mathbb{R}^d)$ we know that by definition, for $|\alpha| = 1$:

$$\|D^{\alpha} f\|^2_{L^2_{\mu}(\mathbb{R}^d)} = \int_{\mathbb{R}^d} |D^{\alpha} f(\mathbf{x}) \cdot D^{\alpha} f(\mathbf{x}) \cdot \mu(\mathbf{x})| d\mathbf{x} < \infty$$

where $d\mathbf{x}$ is the Lebesgue measure. By Minkowski's inequality we know that:

$$\int_{\mathbb{R}^d} |D^{\alpha} f(\mathbf{x}) \cdot D^{\alpha} f(\mathbf{x}) \cdot \mu(\mathbf{x}) \cdot \mu(\mathbf{x})| d\mathbf{x} < \int_{\mathbb{R}^d} |\mu(\mathbf{x})| d\mathbf{x} \int_{\mathbb{R}^d} |D^{\alpha} f(\mathbf{x}) \cdot D^{\alpha} f(\mathbf{x}) \cdot \mu(\mathbf{x})| d\mathbf{x}$$

By definition μ, a probability measure, is L^1 integrable. As such:

$$\int_{\mathbb{R}^d} |D^{\alpha} f(\mathbf{x}) \cdot D^{\alpha} f(\mathbf{x}) \cdot \mu(\mathbf{x}) \cdot \mu(\mathbf{x})| d\mathbf{x} < \infty$$

Let $m(\mathbf{x}) = D^{\alpha} f(\mathbf{x}) \cdot \mu(\mathbf{x})$, by the equation above, $m(\mathbf{x}) \in L^2(\mathbb{R}^d)$. As both $D^{\alpha} f(\mathbf{x})$ (by assumption) and $m(\mathbf{x})$ are L^2 integrable in $\mathbb{R}^d$, we can apply Fubini's Theorem and Plancherel's Theorem straighforwardly such that:

$$\sum_{|\alpha|=1} \|D^{\alpha} f\|^2_{L^2_{\mu}(\mathbb{R}^d)} = \int_{\mathbb{R}^d} \sum_{j=1}^{d} \left| i\boldsymbol{\omega}_j \mathcal{F}(\boldsymbol{\omega}) \cdot \overline{\mathcal{M}(\boldsymbol{\omega}, j)} \right| d\boldsymbol{\omega}$$

where $\mathcal{F}$ is the Fourier transform of f, $i^2 = -1$, and $\boldsymbol{\omega}_j \mathcal{F}(\boldsymbol{\omega})$ is simply the Fourier transform of the derivative indexed by α. $\mathcal{M}(\boldsymbol{\omega}, j)$ is given by

$$\mathcal{M}(\boldsymbol{\omega}, j) = i \left(\boldsymbol{\omega}_j \mathcal{F}(\boldsymbol{\omega}) \right) * \mathcal{P}(\boldsymbol{\omega}) \tag{A.12}$$

where $\mathcal{P}$ is the Fourier transform of the probability measure μ, $\boldsymbol{\omega}_j \mathcal{F}(\boldsymbol{\omega})$ is as before, and * denotes the convolution operator. Substituting $\mathcal{G}(\boldsymbol{\omega}, j) = \boldsymbol{\omega}_j \mathcal{F}(\boldsymbol{\omega})$ we obtain:

$$\begin{aligned} \sum_{|\alpha|=1} \|D^{\alpha} f\|^2_{L^2_{\mu}(\mathbb{R}^d)} &= \int_{\mathbb{R}^d} \sum_{j=1}^{d} \left| (i\bar{i}) \mathcal{G}(\boldsymbol{\omega}, j) \overline{\mathcal{G}(\boldsymbol{\omega}, j) * \mathcal{P}(\boldsymbol{\omega})} \right| d\boldsymbol{\omega} \\ &= \int_{\mathbb{R}^d} \sum_{j=1}^{d} \left| \mathcal{G}(\boldsymbol{\omega}, j) \overline{\mathcal{G}(\boldsymbol{\omega}, j) * \mathcal{P}(\boldsymbol{\omega})} \right| d\boldsymbol{\omega} \end{aligned}$$

This concludes the proof.

■

A.1.4 Regularisation in Regression Models and Autoencoders

In the case of regression the most commonly used loss is the mean-square error.

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = \frac{1}{2}(\mathbf{y} - \mathbf{h}_L(\mathbf{x}))^2$$

In this case, $\mathbf{H}_{L,n}$ is $\mathbf{I}$. As such:

$$R(\mathcal{B}; \mathbf{w}) = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim\mathcal{B}}\left[\sum_{k=0}^{L}\sigma_k^2(\mathrm{Tr}(\mathbf{J}_k(\mathbf{x})^\intercal\mathbf{J}_k(\mathbf{x})))\right] = \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim\mathcal{B}}\left[\sum_{k=0}^{L-1}\sigma_k^2(\|\mathbf{J}_k(\mathbf{x})\|_F^2)\right]$$

This added term corresponds to the trace of the covariance matrix of the outputs $\mathbf{h}_L$ given an input $\mathbf{h}_k$. As such we are penalising the sum of output variances of the approximator; we are penalising the sensitivity of outputs to perturbations in layer k (Webb, 1994; Bishop, 1995).

For ReLU-like activations (ELU, Softplus ...) , because our functions are at *most* linear, we can bound our regularisers using the Jacobian of an equivalent linear network:

$$\sum_{k=0}^{L}\sigma_k^2(\|\mathbf{J}_k(\mathbf{x})\|^2) < \sum_{k=0}^{L}\sigma_k^2(\|\mathbf{J}_k^{\text{linear}}(\mathbf{x})\|^2) = \sum_{k=0}^{L}\sigma_k^2(\|\mathbf{W}_L \dots \mathbf{W}_k\|^2)$$

Where $\mathbf{J}_k^{\text{linear}}(\mathbf{x})$ is the gradient evaluated with no non-linearities in our network. This upper bound is reminiscent of $rank - k$ ridge regression, but here we penalise each sub-network in our network (Kunin et al., 2019). Also note that the regression setting is directly translatable to Auto-Encoders, where the labels are the input data.

A.1.5 Regularisation in Classifiers

In the case of classification, we consider the cross-entropy loss. Recall that we consider our network outputs $\mathbf{h}_L$ to be the pre-softmax of logits of the final layer $\mathbf{L}$. We denote

$\mathbf{p}(\mathbf{x}) = \text{softmax}(\mathbf{h}_L(\mathbf{x}))$. The loss is thus:

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = -\sum_{c=0}^{M} \mathbf{y}_{n,c} \log(\text{softmax}(\mathbf{h}_L(\mathbf{x}))_c) \tag{A.13}$$

where c indexes over the M possible classes of the classification problem. The hessian $\mathbf{H}_L$ in this case is easy to compute and has the form:

$$\mathbf{H}_L(\mathbf{x})_{i,j} = \begin{cases} \mathbf{p}(\mathbf{x})_i(1 - \mathbf{p}(\mathbf{x})_j) & i = j \\ -\mathbf{p}(\mathbf{x})_i \mathbf{p}(\mathbf{x})_j & i \neq j \end{cases} \tag{A.14}$$

As Wei et al. (2020), Sagun et al. (2018), and LeCun et al. (1998a) show, this Hessian is PSD, meaning that $\text{Tr}(\mathbf{J}_k \mathbf{H}_L \mathbf{J}_k^\intercal)$ will be positive, fulfilling the criteria for a valid regulariser.

$$R(\mathcal{B}; \mathbf{w}) = \mathbb{E}_{\mathbf{x}\sim\mathcal{B}} \left[\frac{1}{2} \sum_{k=0}^{L} \sigma_k^2 \sum_{i,j} (\mathbf{H}_L(\mathbf{x}) \circ \mathbf{J}_k(\mathbf{x}) \mathbf{J}_k^\intercal(\mathbf{x}))_{i,j} \right]$$
$$= \mathbb{E}_{\mathbf{x}\sim\mathcal{B}} \left[\frac{1}{2} \sum_{k=0}^{L} \sigma_k^2 \sum_{i,j} (\text{diag}(\mathbf{H}_L(\mathbf{x}))^\intercal \mathbf{J}_k^2(\mathbf{x}))_{i,j} + \frac{1}{2} \sum_{k=0}^{L} \sigma_k^2 \sum_{\forall i,j \ i \neq j} (\mathbf{H}_L(\mathbf{x}) \circ \mathbf{J}_k(\mathbf{x}) \mathbf{J}_k^\intercal(\mathbf{x}))_{i,j} \right]$$

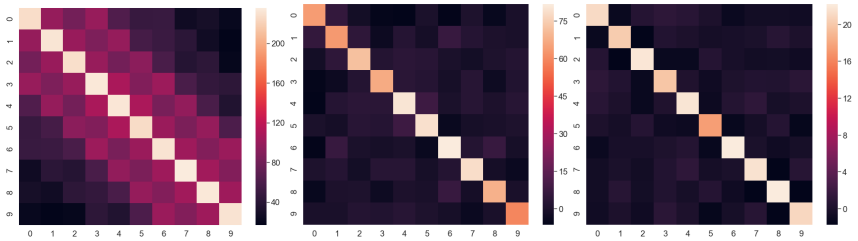
$\text{diag}(\mathbf{H}_L(\mathbf{x}))^\intercal$ is the row vector of the diagonal of $\mathbf{H}_L(\mathbf{x})$. The first equality is due to the fact that $\mathbf{H}_L$ is symmetric and is due to the commutative properties of the trace operator. The final equality is simply the decomposition of the sum of the matrix product into diagonal and off-diagonal elements. For shallow networks, the off-diagonal elements of $\mathbf{J}_k \mathbf{J}_k^\intercal$ are likely to be small and it can be approximated by $\mathbf{J}_k^2$ (Poole et al., 2016; Hauser and Ray, 2017; Farquhar et al., 2020; Aleksziev, 2019). See Figure A.1.1 for a demonstration that the off-diagonal elements of $\mathbf{J}_k^\intercal \mathbf{J}_k$, are negligible for smaller networks. Ignoring these off-diagonal terms, we obtain an added positive term:

$$R(\mathcal{B}; \mathbf{w}) \approx \mathbb{E}_{\mathbf{x}\sim\mathcal{B}} \left[\frac{1}{2} \sum_{k=0}^{L} \sigma_k^2 \sum_{i,j} (\text{diag}(\mathbf{H}_L(\mathbf{x}))^\intercal \mathbf{J}_k^2(\mathbf{x}))_{i,j} \right] \tag{A.15}$$

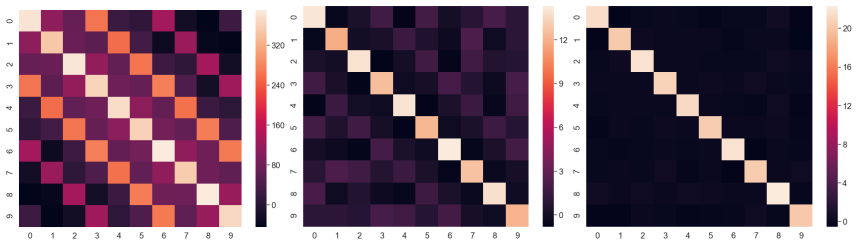
For ReLU-like activations (ELU, Softplus ...), because our functions are at *most* linear,

we can bound our regularisers using the Jacobian of an equivalent linear network:

$$\sum_{k=0}^{L} \sigma_k^2 \sum_{i,j} (\text{diag}(\mathbf{H}_L(\mathbf{x}))^\intercal \mathbf{J}_k(\mathbf{x})^2)_{i,j} < \sum_{k=0}^{L} \sigma_k^2 \sum_{i,j} (\text{diag}(\mathbf{H}_L(\mathbf{x}))^\intercal (\mathbf{W}_L \dots \mathbf{W}_k)^2))_{i,j} \quad \text{(A.16)}$$



(a) SVHN-MLP, k=0 (b) SVHN-MLP, k=1 (c) SVHN-MLP, k=2



(d) CIFAR-CONV,k=0 (e) CIFAR-CONV,k=1 (f) CIFAR-CONV,k=2

Figure A.1.1: Samples of heatmaps of 10 by 10 matrices $\mathbf{J}_k^\intercal \mathbf{J}_k$ (k indexing over layers) for 2-layer MLPs and convolutional networks (CONV) trained to convergence (with no regularisation) on the SVHN and CIFAR10 classification datasets, each with 10 classes. We can clearly see that the diagonal elements of these matrices dominate in all examples, though less so for the data layer.

A.2 Tikhonov Regularisation

Note that because we are penalising the terms of the Sobolev norm associated with the first order derivatives, this constitutes a form of Tikhonov regularisation. Tikhonov regularisation involves adding some regulariser to the loss function, which encodes a notion of 'smoothness' of a function f (Bishop, 1995). As such, by design, regularisers of this form have been shown to have beneficial regularisation properties when used in the training objective of neural networks by smoothing the loss landscape (Girosi and Poggio, 1990; Burger and Neubauer, 2003). If we have a loss of the form $\mathcal{L}(\mathcal{B}; \mathbf{w})$, the Tikhonov regularised loss becomes:

$$\mathcal{L}(\mathcal{B}; \mathbf{w}) + \lambda \| f^\theta \|^2_{\mathcal{H}} \tag{A.17}$$

where f^θ is the function with parameters θ which we are learning and $\| \cdot \|_{\mathcal{H}}$ is the norm or semi-norm in the Hilbert space $\mathcal{H}$ and λ is a (multidimensional) penalty which penalises elements of $\| f^\theta \|^2_{\mathcal{H}}$ unequally, or is data-dependent (Tikhonov and Arsenin, 1977; Bishop, 1995). In our case $\mathcal{H}$ is the Hilbert-Sobolev space $W^{1,2}_\mu(\mathbb{R}^d)$ with norm dictated by Equation (3.7). $R(\cdot)$ penalises the function's semi-norm in this space.

A.3 Measuring Calibration

A neural network classifier gives a prediction $\hat{y}(\mathbf{x})$ with confidence $\hat{p}(\mathbf{x})$ (the probability attributed to that prediction) for a datapoint $\mathbf{x}$. Perfect calibration consists of being as likely to be correct as you are confident:

$$p(\hat{y} = y | \hat{p} = r) = r, \quad \forall r \in [0, 1] \tag{A.18}$$

To see how closely a model approaches perfect calibration, we plot reliability diagrams (Guo et al., 2017; Niculescu-Mizil and Caruana, 2005), which show the accuracy of a model as a function of its confidence over M bins B_m.

$$\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbf{1}(\hat{y}_i = y_i) \tag{A.19}$$

$$\text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i \tag{A.20}$$

We also calculate the Expected Calibration Error (ECE) Naeini et al. (2015), the mean difference between the confidence and accuracy over bins:

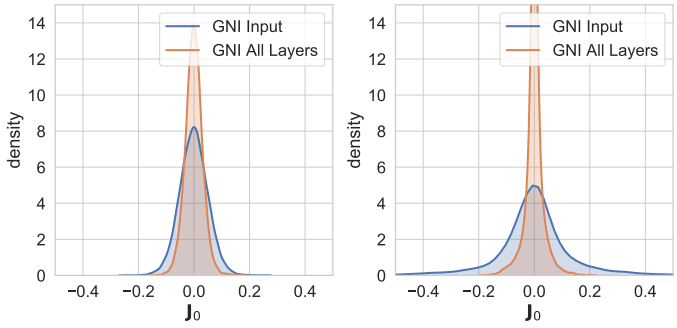
$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{N} |\text{acc}(B_m) - \text{conf}(B_m)| \tag{A.21}$$

However, note that ECE only measures calibration, not refinement. For example, if we have a balanced test set one can trivially obtain ECE ≈ 0 by sampling predictions from a uniform distribution over classes while having very low accuracy.

A.4 Classification Margins

Typically, models with larger classification margins are less sensitive to input perturbations (Sokolić et al., 2017; Jakubovitz and Giryes, 2018; Cohen et al., 2019; Liu et al., 2019; Li et al., 2018). Such margins are the distance in data-space between a point $\mathbf{x}$ and a classifier's decision boundary. Larger margins mean that a classifier associates a larger region centered on a point $\mathbf{x}$ to the same class. Intuitively this means that noise added to $\mathbf{x}$ is still likely to fall within this region, leaving the classifier prediction unchanged. Sokolić et al. (2017) and Jakubovitz and Giryes (2018) define a classification margin M that is the radius of the largest metric ball centered on a point $\mathbf{x}$ to which a classifier assigns $\mathbf{y}$, the true label.

Definition A.4.1 (Classification Margin (Jakubovitz and Giryes (2018))). *Consider a classifier that outputs a correct prediction for the true class A associated with a point $\mathbf{x}$. Then the first order approximation for the l2-norm of the classification margin M, which*



(a) $\mathbf{J}_0$ CIFAR (b) $\mathbf{J}_0$ SVHN

Figure A.4.2: Here we show distribution plots of $\mathbf{J}_0$ for 2-layer MLPs trained on CIFAR10 (a) and SVHN (b) for models trained with no noise (Baseline), models trained with noise on their inputs (GNI Input), models trained with noise on all their layers (GNI All Layers). Noising all layers induces a larger penalisation on the norm of $\mathbf{J}_0$, seen clearly here by the shrinkage to 0 of $\mathbf{J}_0$ for models trained in this manner.

is the minimal perturbation necessary to fool a classifier, is lower bounded by:

$$M(\mathbf{x}) \geq \frac{(\mathbf{h}_L^A(\mathbf{x}) - \mathbf{h}_L^B(\mathbf{x}))}{\sqrt{2}\|\mathbf{J}_0(\mathbf{x}))|_F}. \tag{A.22}$$

We have $\mathbf{h}_L^A(\mathbf{x}) \geq \mathbf{h}_L^B(\mathbf{x})$*, where* $\mathbf{h}_L^A(\mathbf{x})$ *is the* L^{th} *layer activation (pre-softmax) associated with the true class* A*, and* $\mathbf{h}_L^B(\mathbf{x})$ *is the second largest* L^{th} *layer activation.*

Networks that have lower-frequency spectrums and consequently have smaller norms of Jacobians (as established in Section 3.4), will have larger classification margins and will be less sensitive to perturbations. This explains the empirical observations of Rahaman et al. (2019) which showed that functions biased towards lower frequencies are more robust to input perturbations.

What does this entail for GNIs applied to each layer of a network ? We can view the penalisation of the norms of the Jacobians, induced by GNIs for each layer k, as an unpweighted penalisation of $\|\mathbf{J}_0(\mathbf{x})\|_F$. By the chain rule $\mathbf{J}_0$ can be expressed in terms of any of the other network Jacobians $\mathbf{J}_0(\mathbf{x}) = \mathbf{J}_k(\mathbf{x})\frac{\partial \mathbf{h}_k}{\mathbf{x}} \forall k \in [0 \dots L]$. We can write $\|\mathbf{J}_0(\mathbf{x})\|_F = \|\mathbf{J}_k(\mathbf{x})\frac{\partial \mathbf{h}_k}{\mathbf{x}}\|_F \leq \|\mathbf{J}_k(\mathbf{x})\|_F\|\frac{\partial \mathbf{h}_k}{\mathbf{x}}\|_F$. Minimising $\|\mathbf{J}_0(\mathbf{x})\|_F$ is equivalent to minimising $\|\mathbf{J}_k(\mathbf{x})\|_F$ and $\|\frac{\partial \mathbf{h}_k}{\mathbf{x}}\|_F$, and upweighted penalisations of $\|\mathbf{J}_k(\mathbf{x})\|_F$ should translate into a shrinkage of $\|\mathbf{J}_0(\mathbf{x})\|_F$. As such, noising each layer should induce a smaller $\|\mathbf{J}_0(\mathbf{x})\|_F$, and larger classification margins than solely noising data. We support this

empirically in Figure A.4.2.

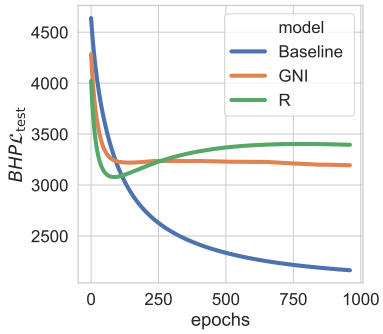
In Figure A.6.6 we confirm that these larger classification translate into a lessened sensitvity to noise.

A.5 Model Capacity

Intuitively one can view lower frequency functions as being 'less complex', and less likely to overfit. This can be visualised in Figure 3.4. A measure of model complexity is given by 'capacity' measures. If we have a model class $\mathcal{H}$, then the capacity assigns a non-negative number to each hypothesis in the model class $\mathcal{M} : \{\mathcal{H}, \mathcal{D}_{train}\} \rightarrow \mathbb{R}^{+}$, where $\mathcal{D}_{train}$ is the training set and a lower capacity is an indicator of better model generalisation (Neyshabur et al., 2017). Generally, deeper and narrower networks induce large capacity models that are likely to overfit and generalise poorly (Zhang et al., 2017). The network Jacobian's spectral norm and Frobenius norm are good approximators of model capacity and are clearly linked to R (Guo et al., 2017; Neyshabur et al., 2017; Neyshabur and Srebro).

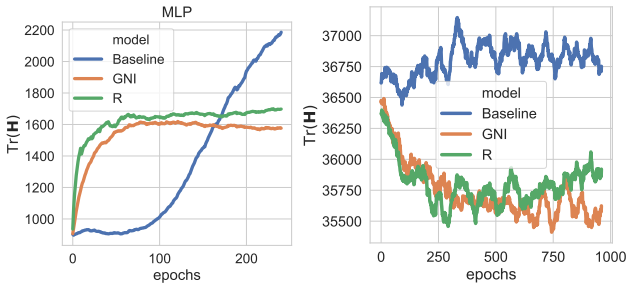
The Frobenius norm of the network Jacobian corresponds to a norm in Sobolev space which is a measure of a network's high-frequency components in the Fourier domain. From this we offer the first theoretical results on why norms of the Jacobian are a good measure of model capacity: as low-frequency functions correspond to smoother functions that are less prone to overfitting, a smaller norm of the Jacobian is thus a measure of a smoother 'less complex' model.

A.6 Additional Results



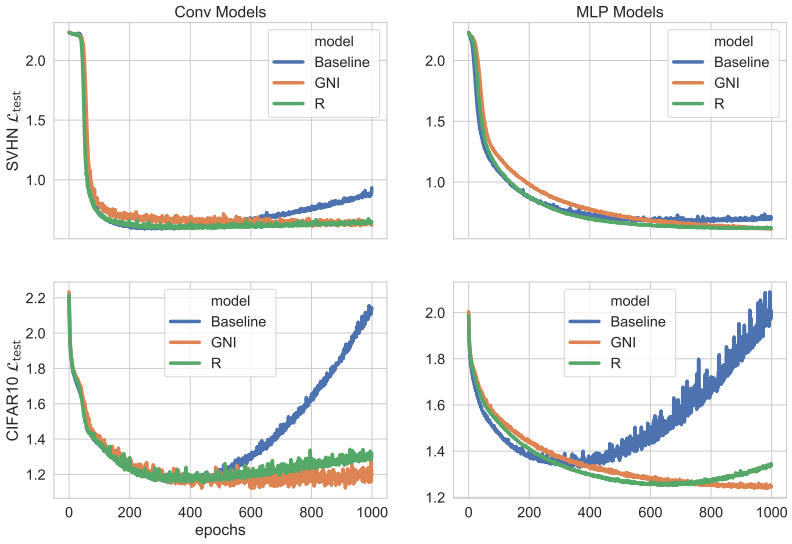
(a) BHP MLP Loss

Figure A.6.3: In Figure (a) we show the test set loss for the regression dataset Boston House Prices (BHP) for 4-layer ELU MLPs trained with $R(\cdot)$ and GNIs for $\sigma^2 = 0.1$. We compare to a non-noised baseline (Baseline). Exp Reg captures much of the effect of noise injections. The test set loss is quasi-identical between Exp Reg and Noise runs which clearly differentiate themselves from Baseline runs.

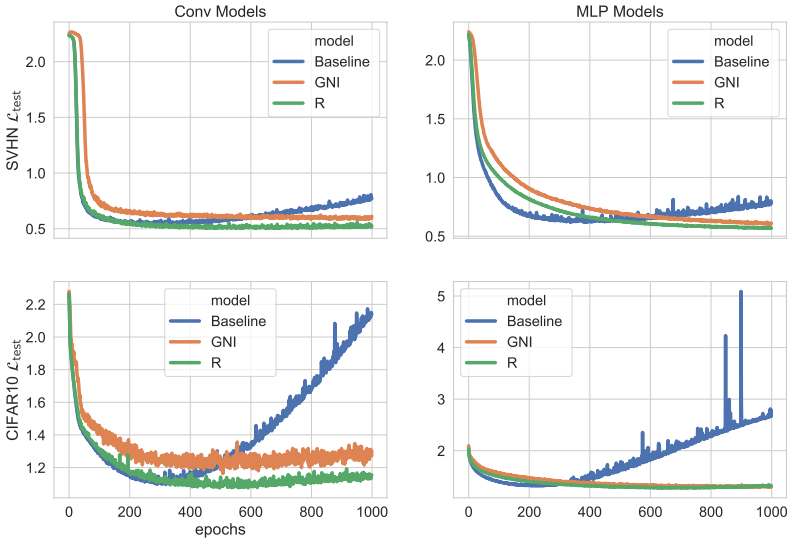


(a) SVHN MLP, $\sigma^2 = 0.1$ (b) BHP MLP $\sigma^2 = 0.1$

Figure A.6.4: Here we use small variance noise injections and show that the $R(\cdot)$ (Exp Reg) in equation (3.11) and (3.14), induces the same trajectory through the loss landscape as GNIs (Noise). We show the trace of the Hessian of neural weights ($H_{i,j} = \frac{\partial \mathcal{L}}{\partial w_i \partial w_j}$) for a smaller 2-layer 32 unit MLP trained on the classification datasets CIFAR10 (a), and SVHN (b), and the regression dataset Boston House Prices (BHP) (c). In all experiments we compare to a non-noised baseline (Baseline). Tr($\mathbf{H}$), which approximates the trajectory of the model weights through the loss landscape, is quasi identical for Exp Reg and Noise and is clearly distinct from Baseline, supporting the fact that the explicit regularisers we have derived are valid. As expected the explicit regulariser and the noised models have smoother trajectories (lower trace) through the loss landscape, except for CIFAR10.

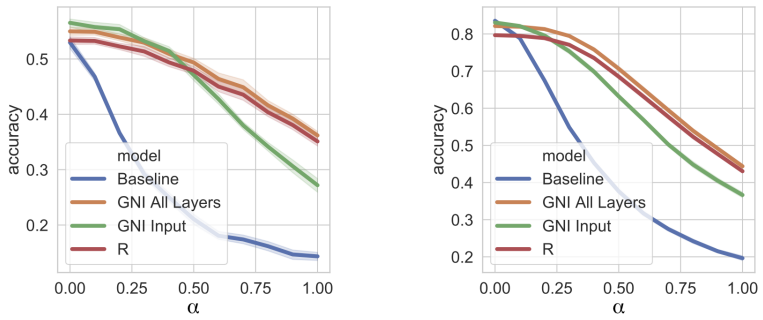


(a) ELU non-linearities, $\sigma^2 = 0.1$



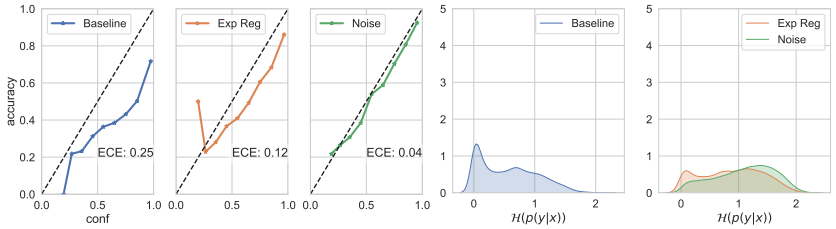
(b) ReLU non-linearities, $\sigma^2 = 0.1$

Figure A.6.5: Illustration of the loss induced by the $R(\cdot)$ for classification detailed in equation (3.14) for convolutional and MLP architectures, and for ReLU and ELU non-linearities. The loss trajectory is quasi-identical to models trained with GNIs and the trajectories are clearly distinct from baselines (Baseline), supporting the fact that the explicit regularisers we have derived are valid.

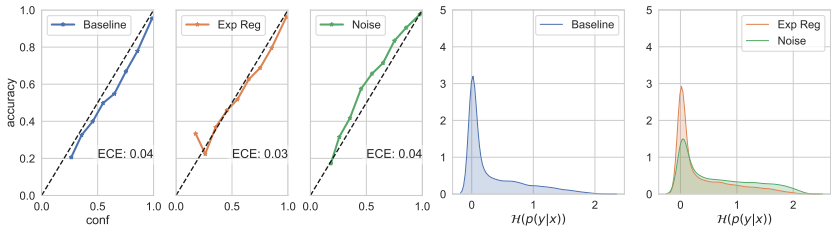


(a) CIFAR (b) SVHN

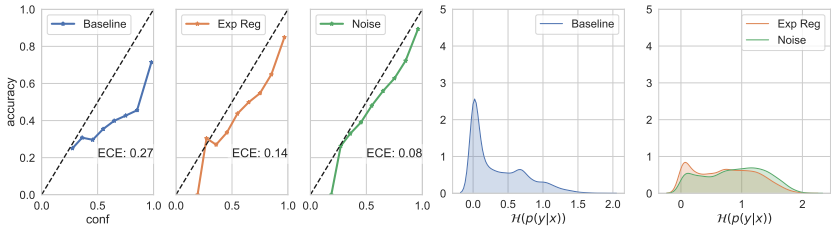
Figure A.6.6: In (a) and (b) a model's sensitivity to noise by adding noise of variance α^2 to data and measuring the resulting model accuracy given this corrupted test data. We show this for 2-layer MLPs trained on CIFAR10 (a) and SVHN (b) for models trained with no noise (Baseline), models trained with noise on their inputs (GNI Input), models trained with noise on all their layers (GNI All Layers), and models trained with the $R(\cdot)$ for classification. Noise added during training has variance $\sigma^2 = 0.1$ and confidence intervals are the standard deviation over batches of size 1024. Models trained with noise on all layers, and those trained with $R(\cdot)$, have the slowest decay of performance as α increases, confirming that such models have larger classification margins.



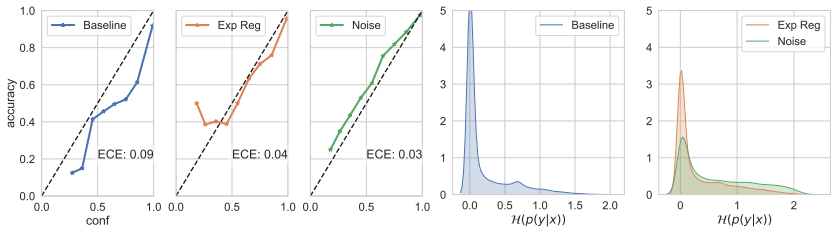
(a) CIFAR10 MLP, $\sigma = 0.1$ (b) CIFAR10 MLP, $\sigma = 0.1$



(c) SVHN MLP, $\sigma = 0.1$ (d) SVHN MLP, $\sigma = 0.1$



(e) CIFAR10 CONV, $\sigma = 0.1$ (f) CIFAR10 CONV, $\sigma = 0.1$



(g) SVHN CONV, $\sigma = 0.1$ (h) SVHN CONV, $\sigma = 0.1$

Figure A.6.7: Illustration of how Gaussian noise (Noise) *additions* improve calibration relative to models trained without noise injections (Baselines) and how $R(\cdot)$ (Exp Reg) also captures some of this improvement in calibration. We include results for MLPs and convolutional networks (CONV) with ELU activations on SVHN and CIFAR10 image datasets. On the left hand side we plot reliability diagrams (Guo et al., 2017; Niculescu-Mizil and Caruana, 2005), which show the accuracy of a model as a function of its confidence over M bins B_m. Models that are perfectly calibrated have their accuracy in a bin match their predicted confidence: this is the dotted line appearing in figures. We also calculate the Expected Calibration Error (ECE) which measures a model's distance to this ideal (see Appendix C for a full description of ECE) (Naeini et al., 2015). Clearly, Noise and Exp Reg models are better calibrated with a lower ECE relative to baselines. This can also be appraised visually in the reliability diagram. The right hand side supports these results. We show density plots of the entropy of model predictions. One-hot, highly confident, predictions induce a peak around 0, which is very prominent in baselines. Both Noise and Exp Reg models smear out predictions, as seen by the greater entropy, meaning that they are more likely to output lower-probability predictions.

A.7 Network Hyperparameters

All networks were trained using stochastic gradient descent with a learning rate of 0.001 and a batch size of 512.

All MLP networks, unless specified otherwise, are 2 hidden layer networks with 512 units per layer.

All convolutional (CONV) networks are 2 hidden layer networks. The first layer has 32 filters, a kernel size of 4, and a stride length of 2. The second layer has 128 filters, a kernel size of 4, and a stride length of 2. The final output layer is a dense layer.

APPENDIX B

APPENDIX: ASYMMETRIC HEAVY TAILS AND IMPLICIT BIAS IN GAUSSIAN NOISE INJECTIONS

B.1 Additional Experimental Results . . . 126

B.2 Cost Functions . . . 129

B.2.1 Mean Square Error . . . 129

B.2.2 Cross Entropy Loss . . . 129

B.3 Other Potential Sources of the Skewness in the Gradient Noise . . . 129

B.4 Overview of the Assumptions . . . 130

B.5 Fractional Differentiation and the Approximation Scheme . . . 130

B.6 Metastability Analysis . . . 134

B.7 Technical Proofs . . . 139

B.7.1 Proof of Lemma 4.1 . . . 139

B.7.2 Proof of Theorem 4.1 . . . 142

B.7.3 Proof of Theorem 4.5 . . . 147

B.7.4 Proof of Theorem B.5.1 . . . 152

B.7.5 Proof of Corollary B.5.3 . . . 155

B.7.6 Proof of Theorem 4.6 . . . 156

The supplementary document is organised as follows.

1. The supplementary document begins first with a presentation of additional experiments that are referenced directly in the main text (Section B.1).

2. We then cover the cost-functions used to train neural networks in Section B.2; and give an overview in Section B.3 of the other potential sources of the implicit effect gradient noise skew which we explored.

3. In Section B.4, we provide an overview of the assumptions we will be making in our analysis. We then describe in Section B.5 the numerical method we use to approximate the drift term $b(\mathbf{w}, \alpha, \theta)$ defined in (4.19).

4. We end with metastability analysis of asymmetric stable processes (Section B.6); followed by the technical proofs of the lemmas, theorems, and corollaries that we present in the main body and the supplementary document of the paper (Section B.7).

Before beginning the supplementary document we make a quick note of network architectures and training hyper-parameters.

Network Architectures Networks were trained using stochastic gradient descent with a learning rate of 0.0003 and batch sizes specified in text. MLP network architectures are specified in text. Convolutional (CONV) networks are 2 hidden layer networks. The first layer has 32 filters, a kernel size of 4, and a stride length of 2. The second layer has 128 filters, a kernel size of 4, and a stride length of 2. The final output layer is a dense layer.

B.1 Additional Experimental Results

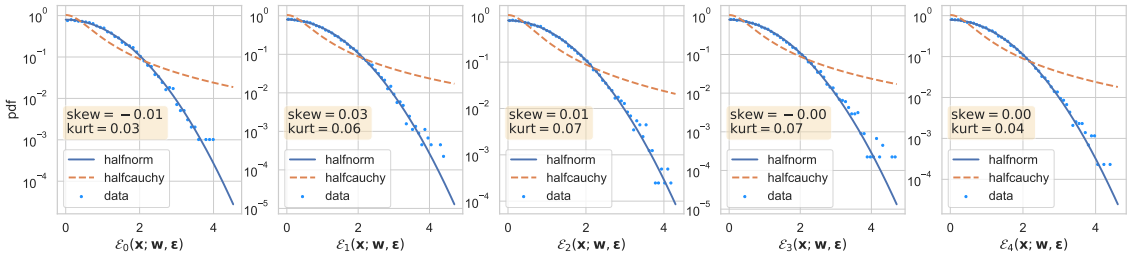


Figure B.1.1: We measure the skewness and kurtosis *at initialisation* of the noise accumulated on network activations at each layer i for a 4-layer 256-unit wide MLP trained to regress the function $\lambda(\mathbf{x}) = \sum_i \sin(2\pi q_i \mathbf{x} + \phi(i))$ with $q_i \in (5, 10, \ldots, 45, 50)$, $\mathbf{x} \in \mathbb{R}$ and experiencing additive-GNIs. We plot the probability density function of positive samples, comparing against half-normal (non-heavy-tailed) and half-Cauchy (heavy-tailed) distributions, where $\mathcal{E}_i(\mathbf{x};\mathbf{w},\boldsymbol{\epsilon})$ is defined in (4.9). Each blue point represents the noise on an individual activation in a layer i for a point $\mathbf{x}$. This noise is Gaussian (low skew and kurtosis) with a p.d.f. that tracks that of a half-normal.

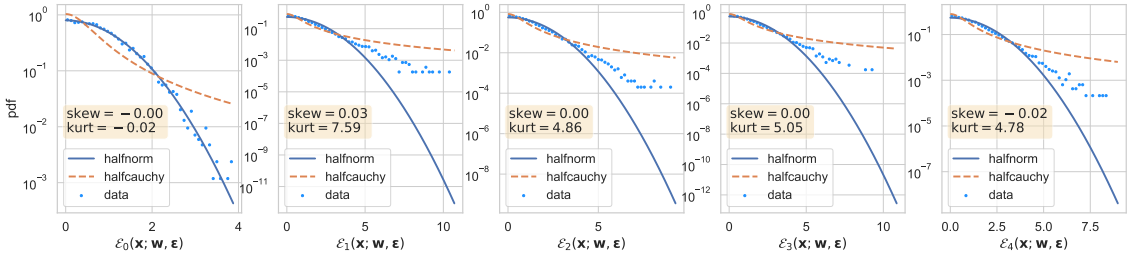


Figure B.1.2: Here we show the same plots as in Figure B.1.1 but for multiplicative-GNIs. The forward pass here experiences *symmetric heavy-tailed noise* for all layers past the data layer.

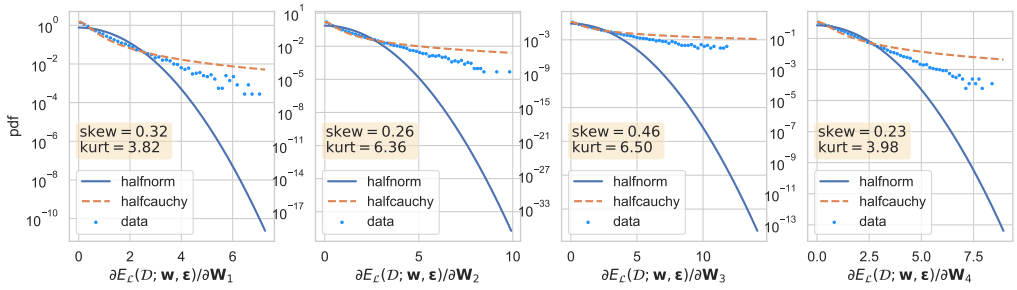
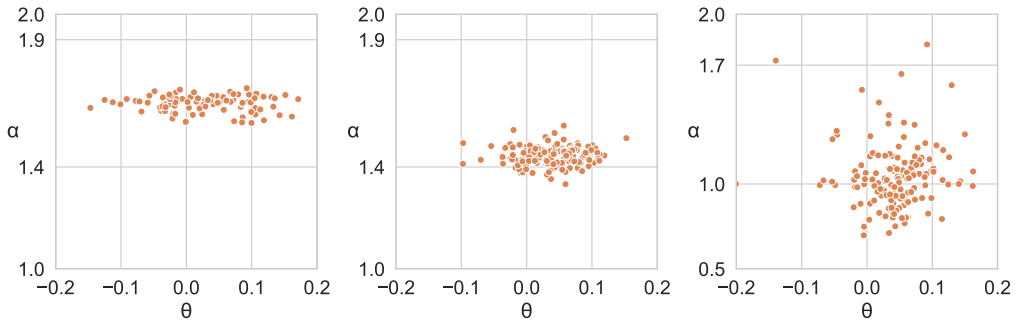


Figure B.1.3: Here we show the same plots as in Figure 4.2 but for multiplicative-GNIs. The gradient noise is *skewed and heavy-tailed*, with a p.d.f. that is more Cauchy-like than Gaussian. The kurtosis decays as the gradients approach the input layer, as predicted by Theorem 4.1.



(a) CIFAR10 (×) (b) SVHN (×) (c) BHP (+)

Figure B.1.4: We model $\nabla E_{\mathcal{L}}(\cdot)$ as being drawn from some α-stable distribution $\mathcal{S}_\alpha$ and estimate the tail-index α and skewness θ using maximum likelihood estimation as in Nolan (2001). We plot the results as a scatter for a batch of size $B = 512$ for CIFAR10 and SVHN and $B = 32$ for Boston House Prices. Additive (+) and multiplicative (×) GNIs have $\sigma^2 = 0.1$

B.2 Cost Functions

B.2.1 Mean Square Error

In the case of regression the most commonly used loss is the mean-square error.

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = (\mathbf{y} - \mathbf{h}_L(\mathbf{x}))^2 \ .$$

B.2.2 Cross Entropy Loss

In the case of classification, we use the cross-entropy loss. If we consider our network outputs $\mathbf{h}_L$ to be the pre-softmax of logits of the final layer then the loss is for a data-label pair $(\mathbf{x}, \mathbf{y})$

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = -\sum_{c=0}^{C} \mathbf{y}_c \log \left(\text{softmax}(\mathbf{h}_L(\mathbf{x}))_c\right) \ , \tag{B.1}$$

where c indexes over the C possible classes of the classification problem.

B.3 Other Potential Sources of the Skewness in the Gradient Noise

The product of correlated random variables can be skewed (Oliveira et al., 2016; Nadarajah and Pogány, 2016). Our first hypothesis was that the skew came from the correlation of $(\partial E_{\mathcal{L}}(\cdot)/\partial h_i^m)$ and $(\partial h_i^m/\partial W_{i,l,j})$. As a test, Figure B.3.5 of the Appendix reproduces Figure 4.2 with linear κ, isolating gradient correlation as a potential source of skew. Gradients are not skewed, demonstrating that the asymmetry stems from non-linear κ.

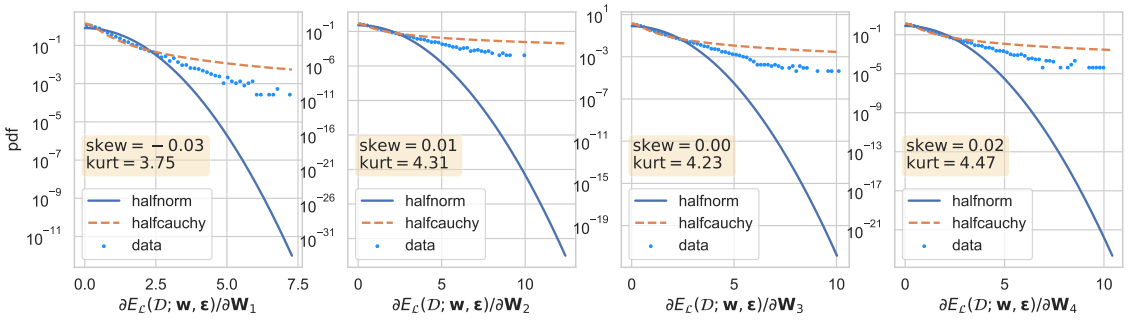


Figure B.3.5: Here we reproduce Figure 4.2 but with no non-linearities. This gradient noise is clearly *heavy-tailed* but *not* skewed.

B.4 Overview of the Assumptions

To avoid obscuring the main take home messages of our theoretical results, we did not present the two assumption required for Theorem 4.6. These two assumptions are properly presented in their respective sections (Sections B.5 and B.7.6), where we first provide the required technical context for defining them in each section. In this section, we will shortly discuss the semantics of these assumptions from a higher-level perspective for the convenience of the reader.

- **Assumption B.5.2.** This assumption is essentially an assumption of the tails of the function $\partial\varphi$, with $\varphi(w) = e^{-\varepsilon^{-\alpha} f(w)}$. In particular, in order to make our approximation scheme (to the fractional derivatives) convergent, this assumption makes sure that outside of a compact region, the function $\partial\varphi$ exponentially decays.

- **Assumption B.7.10.** This assumption enforces a certain structure on the Euler-Maruyama discretisation given in Section 4.4:

$$\tilde{\mathbf{w}}_{n+1} = \tilde{\mathbf{w}}_n + \eta_{n+1} b_{h,K}(\tilde{\mathbf{w}}_n, \alpha, \theta) + \varepsilon \eta_{n+1}^{1/\alpha} \Delta \mathbf{L}_{n+1}^{\alpha,\theta}.$$

  As a first condition, we make sure that the step-sizes are decreasing while their sum is diverging, which is a standard assumption. The second condition is essentially a Lyapunov condition that requires the modified drift b behaves well, so that we can control the weak error of the sample averages by using Panloup (2008). The final condition is similar to the second condition in nature, and requires ergodicity of an SDE defined through the approximate drift $b_{h,K}$, in order to enable us link the weak error to the error induced by the approximation scheme used for the fractional derivatives.

B.5 Fractional Differentiation and the Approximation Scheme

In this section, we provide the details of the Riesz-Feller type fractional derivative $\mathcal{D}^{\alpha-2,-\theta}$, whose definition was omitted in the main document for clarity. We then present the details of the approximation method for the drift term $b(\mathbf{w}, \alpha, \theta)$ defined in (4.19).

The building block of our analysis is a first-order approximation of $\mathcal{D}^{\alpha-2,-\theta}\partial_w\varphi$ for any $\partial_w\varphi \in L^1(\mathbb{R}) \cap \mathcal{C}^4(\mathbb{R})$. We consider the one-dimensional case for simplicity since the Lévy motion we consider has independent coordinates, and the multi-dimensional numerical approximation can be reduced to the one-dimensional case. Assume the tail index $1 < \alpha < 2$ and the skewness parameter satisfies $-1 < \theta < 1$.

When $\theta = 0$, Şimşekli (2017) developed the numerical approximation method for the drift term $b(\mathbf{w}, \alpha, 0)$ by approximately computing the Riesz potential[1] $\mathcal{D}^\gamma$ via the fractional centred difference method provided by Ortigueira (2006); Ortiguera (2006b). It is shown that for any $-1 < \gamma < 0$, we have the following numerical error,

$$\left|\mathcal{D}^\gamma\partial_w\varphi(w) - \Delta^\gamma_{h,K}\partial_w\varphi(w)\right| = \mathcal{O}\left(h^2 + 1/(hK)\right) ,$$

as $h \to 0$, where $K \in \mathbb{N} \cup \{0\}$ is the truncation parameter and $\partial_w\varphi(w)$ satisfies some regularity conditions, and the operator $\Delta^\gamma_{h,K}$ is given by

$$\Delta^\gamma_{h,K} f(w) = \frac{1}{h^\gamma}\sum_{k=-K}^{K} g_{\gamma,k} f(w - kh),$$

for any test function f satisfying some regularity conditions, where

$$g_{\gamma,k} := \frac{(-1)^k\Gamma(\gamma+1)}{\Gamma\left(\frac{\gamma}{2} - k + 1\right)\Gamma\left(\frac{\gamma}{2} + k + 1\right)}.$$

We study the numerical error when approximating the drift term $b(\mathbf{w}, \alpha, \theta)$ with the skewness parameter $-1 < \theta < 1$ and provide the truncation error with a truncation parameter K in Corollary B.5.3. Based on this result, Theorem 4.6 follows, which quantifies the bias induced by α-stable noise on gradient updates using the Euler-Maruyama scheme.

Instead of using the centred difference method to implement the approximation for the $\theta = 0$ case, we tackle the more general $\theta \neq 0$ case by using shifted Grünwald-Letnikov difference operators to approach the left and right fractional derivative respectively. Let us define the parameter $-1 < \gamma := \alpha - 2 < 0$. Then, we can now formally define the

[1]Note that when $\theta = 0$, $\mathcal{D}^{\alpha-2} = \mathcal{D}^{\alpha-2,0}$ recovers the Riesz potential.

Riesz-Feller type fractional derivative operator as follows:

$$\mathcal{D}^{\gamma,-\theta} f(w) := \frac{1}{2\cos(\gamma\pi/2)} \left[(1-\theta)\mathcal{I}_{+}^{-\gamma} f(w) + (1+\theta)\mathcal{I}_{-}^{-\gamma} f(w) \right], \tag{B.2}$$

with

$$\mathcal{I}_{\pm}^{-\gamma} f(w) := \frac{1}{\Gamma(-\gamma)} \int_0^\infty \frac{f(w \pm \xi)}{\xi^{\gamma+1}} d\xi \,. \tag{B.3}$$

Before we proceed, we first introduce difference operators $\mathcal{A}_{h,p}^{\gamma}$ and $\mathcal{B}_{h,q}^{\gamma}$, where p and q are two non-negative integers chosen to be the shifted parameters,

$$\mathcal{A}_{h,p}^{\gamma} f(w) = \frac{1}{h^\gamma} \sum_{k=0}^{\infty} \tilde{g}_{\gamma,k} f(w - (k-p)h), \tag{B.4}$$

$$\mathcal{B}_{h,q}^{\gamma} f(w) = \frac{1}{h^\gamma} \sum_{k=0}^{\infty} \tilde{g}_{\gamma,k} f\left(w + (k-q)h\right). \tag{B.5}$$

Essentially, we defined a forward shifted difference operator $\mathcal{A}_{h,p}^{\gamma}$ to approximate the left fractional derivatives, and a backward shifted difference operator $\mathcal{B}_{h,p}^{\gamma}$ to approximate the right one. The coefficients $\tilde{g}_{\gamma,k} := \frac{(-1)^k \Gamma(-\gamma+k)}{\Gamma(k+1)\Gamma(-\gamma)}$ are from the coefficients of of the power series $(1-z)^\gamma$ with $-1 < \gamma < 0$ and $|z| \leq 1$. For any negative fractional number $-1 < \gamma < 0$ and $|z| \leq 1$, we have

$$(1-z)^\gamma = \sum_{k=0}^{\infty} (-1)^k \binom{-\gamma+k-1}{k} z^k, \tag{B.6}$$

where the binomial coefficient $\binom{-\gamma+k-1}{k}$ is well-defined and the binomial series converges for any complex number $|z| \leq 1$; see e.g. Kroneburg (2011). Indeed, when $-1 < \gamma < 0$, we get

$$\binom{-\gamma+k-1}{k} = \frac{\Gamma(-\gamma+k)}{\Gamma(k+1)\Gamma(-\gamma)}. \tag{B.7}$$

We first present the following first-order approximation result of the fractional derivative $\mathcal{D}^{\gamma,-\theta}$.

Theorem B.5.1. *Let $\mathcal{D}^{\gamma,-\theta}$ denote the fractional derivative for $-1 < \gamma < 0$ and $-1 <$*

$\theta < 1$ as in (B.2). Suppose the function $f \in L^1(\mathbb{R}) \cap \mathcal{C}^4(\mathbb{R})$. Define

$$\Delta_{h,p,q}^{\gamma,-\theta} f(w) = \frac{1}{2\cos(\gamma\pi/2)} \left[(1+\theta)\mathcal{A}_{h,p}^{\gamma} f(w) + (1-\theta)\mathcal{B}_{h,q}^{\gamma} f(w) \right]. \tag{B.8}$$

Then $\Delta_{h,p,q}^{\gamma,-\theta} f(w)$ is an approximation of $\mathcal{D}^{\gamma,-\theta} f(w)$ with the first-order accuracy:

$$\begin{aligned} &\left| \mathcal{D}^{\gamma,-\theta} f(w) - \Delta_{h,p,q}^{\gamma,-\theta} f(w) \right| \\ &\quad \leq \left[|p-q| + |\theta|(p+q-\gamma) + \left| \tan\left(\frac{\gamma\pi}{2}\right) \right| (p+q-\gamma+|\theta||p-q|) \right] \frac{C}{4\pi(|\gamma|+2)} h + \mathcal{O}\left(h^2\right), \end{aligned} \tag{B.9}$$

as $h \to 0$, uniformly for all $w \in \mathbb{R}$, where $C > 0$ is a constant that may depend on f and $\mathcal{O}(\cdot)$ hides the dependence on p, q and γ.

Next, we provide an error bound for numerically computing the drift term $b(\mathbf{w}, \alpha, \theta)$ by truncating the approximation series in Theorem B.5.1 as follows. Let us first define the operators $\mathcal{A}_{h,p,K}^{\gamma}$ and $\mathcal{B}_{h,q,K}^{\gamma}$:

$$\mathcal{A}_{h,p,K}^{\gamma} f(w) := \frac{1}{h^{\gamma}} \sum_{k=0}^{K} \tilde{g}_{\gamma,k} f\left(w - (k-p)h\right), \tag{B.10}$$

$$\mathcal{B}_{h,q,K}^{\gamma} f(w) := \frac{1}{h^{\gamma}} \sum_{k=0}^{K} \tilde{g}_{\gamma,k} f\left(w + (k-q)h\right), \tag{B.11}$$

with $\tilde{g}_{\gamma,k} := \frac{(-1)^k \Gamma(-\gamma+k)}{\Gamma(k+1)\Gamma(-\gamma)}$, and $K \in \mathbb{N} \cup \{0\}$.

Before we state the next result, let us first introduce the following assumption.

Assumption B.5.2. *Suppose the function $\partial_w \varphi \in L^1(\mathbb{R}) \cap \mathcal{C}^4(\mathbb{R})$. In addition, there exist constants $C_p, C_q > 0$ satisfying*

$$|\partial_w \varphi(w - |k-p|h)| \leq C_p e^{-|k-p|h}, \quad |\partial_w \varphi(w + |k-q|h)| \leq C_q e^{-|k-q|h}, \tag{B.12}$$

and $\min\{|k-p|, |k-q|\} > K$ for the constant $K \in \mathbb{N} \cup \{0\}$.

We have the following result.

Corollary B.5.3. *Suppose Assumption B.5.2 holds for $\partial_w \varphi$, and recall the truncated series $\mathcal{A}_{h,p,K}^{\gamma}$ and $\mathcal{B}_{h,q,K}^{\gamma}$ with $K \in \mathbb{N} \cup \{0\}$ defined in (B.10) and (B.11). Let us also*

define the operator:

$$\Delta^{\gamma,-\theta}_{h,p,q,K} = \frac{1}{2\cos(\gamma\pi/2)}\left[(1-\theta)\mathcal{B}^{\gamma}_{h,q,K} + (1+\theta)\mathcal{A}^{\gamma}_{h,p,K}\right]. \tag{B.13}$$

Then the truncation error is bounded in first-order accuracy as follows,

$$\begin{aligned}&\left|\mathcal{D}^{\gamma,-\theta}\partial_w\varphi(w) - \Delta^{\gamma,-\theta}_{h,p,q,K}\partial_w\varphi(w)\right| \\ &\quad\leq \frac{C}{4\pi(|\gamma|+2)}\left[|p-q| + |\theta|(p+q-\gamma) + \left|\tan\left(\frac{\gamma\pi}{2}\right)\right|(p+q-\gamma+|\theta||p-q|)\right]h \\ &\qquad + \left((1+\theta)C_p + (1-\theta)C_q\right)\frac{1}{hK} + \mathcal{O}(h^2),\end{aligned} \tag{B.14}$$

where $C, C_p, C_q > 0$ are constants that may depend on $\partial_w\varphi$ and $\mathcal{O}(\cdot)$ hides the dependence on p and q.

In particular, by taking $p = q = 0$, Corollary B.5.3 implies that

$$\begin{aligned}&\left|\mathcal{D}^{\gamma,-\theta}\partial_w\varphi(w) - \Delta^{\gamma,-\theta}_{h,p=0,q=0,K}\partial_w\varphi(w)\right| \\ &\quad\leq \frac{C}{4\pi}\left(|\theta| + \left|\tan\left(\frac{\gamma\pi}{2}\right)\right|\right)h + \left((1+\theta)C_{p=0} + (1-\theta)C_{q=0}\right)\frac{1}{hK} + \mathcal{O}(h^2),\end{aligned} \tag{B.15}$$

where $\Delta^{\gamma,-\theta}_{h,K} = \Delta^{\gamma,-\theta}_{h,p=0,q=0,K}$.

Corollary B.5.3 implies that one can approximate $\mathcal{D}^{\gamma,-\theta}$ by the truncated $\Delta^{\gamma,-\theta}_{h,K}$ instead of $\Delta^{\gamma,-\theta}_{h}$. Based on this result, we are able to quantify in Theorem 4.6 the bias induced when implementing Euler-Maruyama scheme to approximate the expectation of a test function g with respect to the target distribution π, where $\nu(g) = \int g(\mathbf{w})\pi(d\mathbf{w})$.

B.6 Metastability Analysis

In this section, we will focus on the metastability properties of the process

$$dw_t = -\nabla_w f(w_t)dt + \varepsilon dL^{\alpha,\theta}_t. \tag{B.16}$$

We will be interested in the first exit time, which is, roughly speaking, the expected time required for the process to exit a neighborhood of a local minimum. We will summarise

the related theoretical results, which show that the first exit time behaviour of systems driven by asymmetric stable processes are similar to the ones of driven by symmetric stable processes. This informally implies that the process will quickly escape from narrow minima regions and will spend more time (in fact will get stuck) in wide minima regions. In this section, we make this argument rigorous.

For simplicity of the presentation, we consider the one-dimensional case where $L_t^{\alpha,\theta}$ is an asymmetric α-stable Lévy process with Lévy measure

$$\nu(dy) = \left(\frac{1-\theta}{2} c_\alpha 1_{y<0} + \frac{1+\theta}{2} c_\alpha 1_{y>0}\right) \frac{dy}{|y|^{1+\alpha}}, \tag{B.17}$$

where $\theta \in (-1,1)$ and $\alpha \in (0,2)$ and $c_\alpha := \frac{\alpha}{\Gamma(1-\alpha)\cos(\pi\alpha/2)}$. Then, the left and right tails of the Lévy measure are given by

$$H_-(-u) := \int_{(-\infty,-u)} \nu(dy) = \frac{1-\theta}{2} C_\alpha u^{-\alpha},$$
$$H_+(u) := \int_{(u,+\infty)} \nu(dy) = \frac{1+\theta}{2} C_\alpha u^{-\alpha},$$

where $C_\alpha := \frac{1-\alpha}{\Gamma(2-\alpha)\cos(\pi\alpha/2)}$, and

$$H(u) := H_-(-u) + H_+(u) = C_\alpha u^{-\alpha}, \qquad \text{for any } u > 0.$$

Let us assume that the function $w \mapsto f(w)$ satisfies the following conditions:

Assumption B.6.4. *(i) $f \in \mathcal{C}^1(\mathbb{R}) \cap \mathcal{C}^3([-K,K])$ for some $K > 0$;*

(ii) f has exactly n local minima m_i, $1 \le i \le n$ and $n-1$ local maxima s_i, $1 \le i \le n-1$, enumerated in increasing order with $s_0 = -\infty$ and $s_n = +\infty$:

$$-\infty < m_1 < s_1 < m_2 < \cdots < s_{n-1} < m_n < +\infty. \tag{B.18}$$

All extrema of f are non-degenerate, i.e. $\partial_w^2 f(m_i) > 0$, $1 \le i \le n$, and $\partial_w^2 f(s_i) < 0$, $1 \le i \le n-1$.

(iii) $|\partial_w f(w)| > c_1 |w|^{1+c_2}$ as $w \to \pm\infty$ for some $c_1, c_2 > 0$.

First, we consider the first exit time from a single well. For $\varepsilon > 0$ and $\gamma > 0$, define

$$\Omega_\varepsilon^i := [s_{i-1} + 2\varepsilon^\gamma, s_i - 2\varepsilon^\gamma], \tag{B.19}$$

with the convention that $\Omega_\varepsilon^1 := (-\infty, s_1 - 2\varepsilon^\gamma]$ and $\Omega_\varepsilon^n := [s_{n-1} + 2\varepsilon^\gamma, +\infty)$. The first exit time from the i-th well is defined as

$$\sigma^i(\varepsilon;\theta) := \inf\{t \geq 0 : w_t \notin [s_{i-1} + \varepsilon^\gamma, s_i - \varepsilon^\gamma]\}, \tag{B.20}$$

for $i = 1, \ldots, n$. Let us also define

$$\lambda^i(\varepsilon;\theta) := \frac{1-\theta}{2} C_\alpha \left| \frac{s_{i-1} - m_i}{\varepsilon} \right|^{-\alpha} + \frac{1+\theta}{2} C_\alpha \left| \frac{s_i - m_i}{\varepsilon} \right|^{-\alpha}, \tag{B.21}$$

for $i = 1, \ldots, n$. We have the following first exit time result from Imkeller and Pavlyukevich (2008).

Proposition B.6.5 (Proposition 3.1. in Imkeller and Pavlyukevich (2008))**.** *There exists* $\gamma_0 > 0$ *such that for any* $0 < \gamma \leq \gamma_0$, $i = 1, 2, \ldots, n$,

$$\lambda^i(\varepsilon;\theta)\sigma^i(\varepsilon;\theta) \to \exp(1), \qquad \textit{in distribution as } \varepsilon \to 0, \tag{B.22}$$

where $\exp(1)$ *denotes the exponential distribution with mean* 1*, and*

$$\lim_{\varepsilon \to 0} \mathbb{E}_w \left[\lambda^i(\varepsilon;\theta)\sigma^i(\varepsilon;\theta)\right] = 1, \tag{B.23}$$

where the limit holds uniformly over $w \in \Omega_\varepsilon^i$.

The above result implies that as $\varepsilon \to 0$,

$$\mathbb{E}_w \left[\sigma^i(\varepsilon;\theta)\right] \sim \left(\frac{1-\theta}{2} C_\alpha \left| s_{i-1} - m_i \right|^{-\alpha} + \frac{1+\theta}{2} C_\alpha \left| s_i - m_i \right|^{-\alpha} \right)^{-1} \varepsilon^{-\alpha}. \tag{B.24}$$

If $|s_i - m_i| > |s_{i-1} - m_i|$, i.e. the i-th well is asymmetric and the local minimum m_i is closer to the saddle point on the left s_{i-1} than the saddle point on the right s_i, then $\lambda^i(\varepsilon;\theta) < \lambda^i(\varepsilon;0)$ and $\mathbb{E}_w[\sigma^i(\varepsilon;\theta)] > \mathbb{E}_w[\sigma^i(\varepsilon;0)]$ for positive θ and $\lambda^i(\varepsilon;\theta) > \lambda^i(\varepsilon;0)$ and

$\mathbb{E}_w[\sigma^i(\varepsilon;\theta)] < \mathbb{E}_w[\sigma^i(\varepsilon;0)]$ for negative θ. Similarly, if $|s_i - m_i| < |s_{i-1} - m_i|$, i.e. the i-th well is asymmetric and the local minimum m_i is closer to the saddle point on the right s_i than the saddle point on the left s_{i-1}, then $\lambda^i(\varepsilon;\theta) > \lambda^i(\varepsilon;0)$ and $\mathbb{E}_w[\sigma^i(\varepsilon;\theta)] < \mathbb{E}_w[\sigma^i(\varepsilon;0)]$ for positive θ and $\lambda^i(\varepsilon;\theta) < \lambda^i(\varepsilon;0)$ and $\mathbb{E}_w[\sigma^i(\varepsilon;\theta)] > \mathbb{E}_w[\sigma^i(\varepsilon;0)]$ for negative θ. The intuition is that when the well is asymmetric, the dynamics can exit the well faster when there is a skewness θ towards the the saddle point closer to the minimum of the well.

Next, we consider transitions between the wells. For any $0 < \Delta < \Delta_0 := \min_{1\le i\le n}\{|m_i - s_{i-1}|, |m_i - s_i|\}$ and $w \in \mathbb{R}$ denote $B_\Delta(w) := \{v : |w - v| \le \Delta\}$. Define

$$\tau^i(\varepsilon;\theta) := \inf\{t \ge 0 : w_t \in \cup_{k\neq i} B_\Delta(m_k)\}. \tag{B.25}$$

Then, we have the following result about transitions between the wells from Imkeller and Pavlyukevich (2008).

Proposition B.6.6 (Proposition 4.3. in Imkeller and Pavlyukevich (2008))**.** *For any* $0 < \Delta < \Delta_0$ *and* $j \neq i$

$$\lim_{\varepsilon\to 0} \mathbb{P}_w\left(w_{\tau^i(\varepsilon;\theta)} \in B_\Delta(m_j)\right) = \frac{q_{ij}}{q_i}, \tag{B.26}$$

uniformly for $w \in B_\Delta(m_i)$, $i = 1,\dots,n$, *where*

$$q_{ij} = \left(\frac{1-\theta}{2}1_{j<i} + \frac{1+\theta}{2}1_{j>i}\right)\cdot\left||s_{j-1} - m_i|^{-\alpha} - |s_j - m_i|^{-\alpha}\right|, \quad i \neq j, \tag{B.27}$$

$$-q_{ii} = q_i = \sum_{j\neq i} q_{ij} = \frac{1-\theta}{2}|s_{i-1} - m_i|^{-\alpha} + \frac{1+\theta}{2}|s_i - m_i|^{-\alpha}. \tag{B.28}$$

From (B.26)-(B.28), we can compute that

$$\frac{q_{ij}}{q_i} = \begin{cases} \frac{\left||s_{j-1}-m_i|^{-\alpha} - |s_j - m_i|^{-\alpha}\right|}{\frac{1-\theta}{1+\theta}|s_{i-1}-m_i|^{-\alpha} + |s_i - m_i|^{-\alpha}} & \text{if } j > i, \\ \frac{\left||s_{j-1}-m_i|^{-\alpha} - |s_j - m_i|^{-\alpha}\right|}{|s_{i-1}-m_i|^{-\alpha} + \frac{1+\theta}{1-\theta}|s_i - m_i|^{-\alpha}} & \text{if } j < i. \end{cases} \tag{B.29}$$

Therefore, q_{ij}/q_i is increasing in θ for $j > i$ and decreasing in θ for $j < i$. This is consistent with the intuition that when $\theta > 0$, it is more likely for the dynamics to transit to a well on the right side, and when $\theta < 0$, it is more likely for the dynamics to transit to a well

on the left side.

Next, we consider the following metastability result due to Theorem 1.1. in Imkeller and Pavlyukevich (2008). It describes the metastability phenomenon, which basically says that there exists a time scale under which the system behaves like a continuous time Markov process with a finite state space consisting of values in the set of stable attractors.

Theorem B.6.7 (Theorem 1.1. in Imkeller and Pavlyukevich (2008))**.** *If $w_0 = w \in (s_{i-1}, s_i)$ for some $i = 1, 2, \dots, n$, then for any $t > 0$, in the sense of finite-dimensional distributions,*

$$w_{t/H(1/\varepsilon)} \to Y_t(m_i), \qquad \text{as } \varepsilon \to 0, \tag{B.30}$$

where w_t is defined in (B.16) and $H(1/\varepsilon) = C_\alpha \varepsilon^\alpha$, where $Y_t(m_i)$ that starts at m_i is a continuous-time Markov process on a finite states space $\{m_1, \dots, m_n\}$ with the infinitesimal generator $Q = (q_{ij})_{i,j=1}^n$, where q_{ij} is defined in (B.27).

The Markov process $Y_t(m_i)$ admits a unique invariant distribution π satisfying $Q^T\pi = 0$. In the case of double well, i.e. $n = 2$ and $m_1 < s_1 = 0 < m_2$ separated by a local maximum at $s_1 = 0$, where without loss of generality we assume that $m_2 > |m_1|$, i.e. the second local minimum lies in a wider valley. A simple calculation yields that

$$q_{12} = \frac{1+\theta}{2}\frac{1}{|m_1|^\alpha} = -q_{11}, \quad \text{and} \quad q_{21} = \frac{1-\theta}{2}\frac{1}{m_2^\alpha} = -q_{22}, \tag{B.31}$$

so that it follows from $Q^T\pi = 0$ and $\pi_1 + \pi_2 = 1$ that

$$\pi_1 = \frac{(1+\theta)^{-1}|m_1|^\alpha}{(1+\theta)^{-1}|m_1|^\alpha + (1-\theta)^{-1}m_2^\alpha}, \tag{B.32}$$

$$\pi_2 = \frac{(1-\theta)^{-1}m_2^\alpha}{(1+\theta)^{-1}|m_1|^\alpha + (1-\theta)^{-1}m_2^\alpha}. \tag{B.33}$$

In particular, the ratio $\frac{\pi_2}{\pi_1} = \frac{1+\theta}{1-\theta}\left(\frac{m_2}{|m_1|}\right)^\alpha$ is increasing in $\frac{m_2}{|m_1|}$ and θ. That reveals that in the equilibrium the process will spend more time in the second valley if the second valley is wide and there is a drift towards to the right. In the symmetric case, i.e. $\theta = 0$, $\pi_2 > \pi_1$ since $m_2 > |m_1|$ so that in the equilibrium the process spends more time in the wider valley. In the asymmetric case, i.e. $\theta \neq 0$, if there is a strong skewness towards the left,

i.e. $\theta < 0$ and $|\theta|$ is large, then in the equilibrium the process may spend more time in the narrower valley. Indeed $\pi_2 > \pi_1$ if and only if $\theta > \frac{|m_1|^\alpha - m_2^\alpha}{|m_1|^\alpha + m_2^\alpha}$.

B.7 Technical Proofs

B.7.1 Proof of Lemma 4.1

Before we proceed to the proof of Lemma 4.1 we present some intermediary results that are required for the proof.

Definition B.7.1. *(Asymptotic order of magnitude) A positive sequence a_m is of the same order of magnitude as another positive sequence b_m ($a_m \asymp b_m$, i.e. 'asymptotically equivalent') if there exist some $c, C > 0$ such that: $c \leq \frac{a_m}{b_m} \leq C$ for any $m \in \mathbb{N}$.*

Lemma B.1 (Lemma A.1 in (Vladimirova et al., 2019))**.** *Let X be a normal random variable such that $X \sim \mathcal{N}(0, \sigma^2)$. Then the following asymptotic equivalence holds*

$$\|X\|_m \asymp \sqrt{m}.$$

We know that the centering of variables does not change their tail properties (Vershynin, 2018; Kuchibhotla and Chakrabortty, 2018). As such Lemma B.1 also applies to $X \sim \mathcal{N}(\mu, \sigma^2)$, as $\|X\| \asymp \|X - \mu\| \asymp \sqrt{m}$.

Lemma B.2 (Lemma 3.1 of Vladimirova et al. (2019))**.** *Let $\kappa : \mathbb{R} \to \mathbb{R}$ be a non-linearity that obeys the extended envelope property. And let X be a variable for which $\|X_+\|_m \asymp \|X_-\|_m$ where X_- and X_+ denote the left and right tail of the variable respectively [2]. Then we have:*

$$\|\kappa(X)\|_m \asymp \|X\|_m\,, \qquad \text{for any } m \geq 1\,. \tag{B.34}$$

Lemma B.3. *Let $X_1, \ldots, X_N$ be variables that each obeys $\|X_i\|_m \lesssim m^r, p \in R, i =$*

[2] We weaken Vladimirova et al. (2019)'s requirement for X to be symmetric as the proof they give still holds here.

$1, \dots, N$ *and* $(W_i, \dots, W_N) \in \mathbb{R}^N$.

$$\left\| \sum_{i=1}^{N} W_i X_i \right\|_m \lesssim m^r \,.$$

Proof of Lemma B.3. By Minkowski's inequality we have that

$$\left\| \sum_{i=1}^{N} W_i X_i \right\|_m \leq \sum_{i=1}^{N} \|W_i X_i\|_m \leq \sum_{l=1}^{N} |W_i A_i| m^r, \ (A_1, \dots, A_N) \in \mathbb{R}^N$$
$$\Leftrightarrow \left\| \sum_{i=1}^{N} W_i X_i \right\|_m \lesssim m^r \,.$$

The A_i here are constants that upper bound the asymptotics of each norm $\|X_i\|_m$ in the sum. ■

Proof of Lemma 4.1. **Additive Noise.** Consider first the noised data, $\widetilde{\mathbf{h}}_0(\mathbf{x}) = \mathbf{x} + \boldsymbol{\epsilon}_0, \boldsymbol{\epsilon}_0 \sim \mathcal{N}(0, \sigma_0^2)$. As Lemma B.1 shows, Gaussian random variables have an m^{th} norm that is asymptotically equivalent to $\sqrt{m}$,

$$\left\| \widetilde{h}_{0,l}(\mathbf{x}) \right\|_m \asymp \sqrt{m}, \qquad \text{for any } l = 1, \dots, n_0 \,,$$

where n_0 is the dimensionality of data.

Let us now assume that $\left\| \widetilde{h}_{i,l}(\mathbf{x}) \right\|_m \lesssim \sqrt{m}$, for any $l = 1, \dots, n_i$, for some layer i. The pre-non-linearity at this layer is given by $\tilde{\mathbf{g}}_i = \mathbf{W}_{i+1} \widetilde{\mathbf{h}}_i$. The j^{th} element of $\tilde{\mathbf{g}}_i$ is defined as a sum,

$$g_{i,j}(\mathbf{x}) = \sum_{l=1}^{n_i} W_{i+1,l,j} \widetilde{h}_{i,l}(\mathbf{x}) \,,$$

where $W_{i+1,l,j}$ is the weight that maps from the l^{th} neuron in layer i to the j^{th} in layer $i+1$. By Lemma B.3,

$$\|g_{i,j}(\mathbf{x})\|_m \lesssim \sqrt{m}, \qquad m = 1, \dots, n_i \,.$$

As such if we assume the non-linearities ϕ at each layer obey the extended envelope

property, then we have by Lemma B.2:

$$\begin{aligned}\|\phi(g_{i,j}(\mathbf{x}))\|_m = \left\|\widehat{h}_{i+1,j}(\mathbf{x})\right\|_m &\asymp \|\kappa(g_{i,j}(\mathbf{x}))\|_m \\ \Leftrightarrow \left\|\widehat{h}_{i+1,j}(\mathbf{x})\right\|_m &\lesssim \sqrt{m}, \qquad j = 1, \dots, n_{i+1}\,.\end{aligned}$$

Note that $\widetilde{\mathbf{h}}_{i+1} = \widehat{\mathbf{h}}_{i+1} + \boldsymbol{\epsilon}_{i+1}, \boldsymbol{\epsilon}_{i+1} \sim \mathcal{N}(0, \sigma_{i+1}^2)$. By Lemma B.3, once again $\|\widetilde{h}_{i+1,j}(\mathbf{x})\|_m \lesssim \sqrt{m}$, is Gaussian in its tails. By recursion, with $\widetilde{\mathbf{h}}_0$ as the base case, we have that

$$\left\|\widetilde{h}_{i,l}(\mathbf{x})\right\|_m \lesssim \sqrt{m}, \qquad \text{for any } m \geq 1;\ i = 1, \dots, L-1;\ l = 1, \dots, n_i\,.$$

Multiplicative Noise. Consider first the noised data, $\widetilde{\mathbf{h}}_0(\mathbf{x}) = \mathbf{x} \circ \boldsymbol{\epsilon}_0, \boldsymbol{\epsilon}_0 \sim \mathcal{N}(1, \sigma_0^2)$. As Lemma B.1 shows, Gaussian random variables have an m^{th} norm that is asymptotically equivalent to $\sqrt{m}$,

$$\left\|\widetilde{h}_{0,l}(\mathbf{x})\right\|_m \asymp \sqrt{m}, \qquad \text{for any } l = 1, \dots, n_0\,,$$

where n_0 is the dimensionality of data.

Let us now assume that $\left\|\widetilde{h}_{i,l}(\mathbf{x})\right\|_m \lesssim m^r,\ \forall l = 1, \dots, n_i$, for some layer i. The pre-non-linearity at this layer is given by $\tilde{\mathbf{g}}_i = \mathbf{W}_{i+1}\widetilde{\mathbf{h}}_i$. The j^{th} element of $\tilde{\mathbf{g}}_i$ is defined as a sum,

$$g_{i,j}(\mathbf{x}) = \sum_{l=1}^{n_i} W_{i+1,l,j}\widetilde{h}_{i,l}(\mathbf{x})\,,$$

where $W_{i+1,l,j}$ is the weight that maps from the l^{th} neuron in layer i to the j^{th} in layer $i+1$. By Lemma B.3,

$$\|g_{i,j}(\mathbf{x})\|_m \lesssim m^r, \qquad m = 1, \dots, n_i\,.$$

As such if we assume the non-linearities ϕ at each layer obey the extended envelope

property, then we have by Lemma B.2:

$$\begin{aligned}\|\phi(g_{i,j}(\mathbf{x}))\|_m = \left\|\widehat{h}_{i+1,j}(\mathbf{x})\right\|_m &\asymp \|\kappa(g_{i,j}(\mathbf{x}))\|_m \\ \Leftrightarrow \left\|\widehat{h}_{i+1,j}(\mathbf{x})\right\|_m &\lesssim m^r, \qquad j = 1, \dots, n_{i+1}\,.\end{aligned}$$

Note that $\widetilde{\mathbf{h}}_{i+1} = \widehat{\mathbf{h}}_{i+1} \circ \boldsymbol{\epsilon}_{i+1}, \boldsymbol{\epsilon}_{i+1} \sim \mathcal{N}(1, \sigma^2_{i+1})$. By Hölder's inequality we have that,

$$\left\|\widetilde{\mathbf{h}}_{i+1,j}(\mathbf{x})\right\|_m \lesssim m^{r+\frac{1}{2}}, \qquad j = 1, \dots, n_{i+1}\,.$$

By recursion, with $\widetilde{\mathbf{h}}_0$ as the base case, we have that

$$\left\|\widetilde{h}_{i,l}(\mathbf{x})\right\|_m \lesssim m^{\frac{i+1}{2}}, \qquad \text{for any } m \geq 1;\ i = 1, \dots, L-1;\ l = 1, \dots, n_i\,.$$

■

B.7.2 Proof of Theorem 4.1

Before we proceed to the proof of Theorem 4.1 we present some intermediary results that are required for the proof.

Lemma B.4. *Let X be a bounded random variable such that $|X| \leq C$, then X is sub-Weibull with parameter $\theta = 0$,*

$$\|X\|_m \lesssim m^0, \qquad \textit{for every } m \geq 1. \tag{B.35}$$

Proof of Lemma B.4. The moments of X obey

$$\mathbb{E}\left[|X|^m\right] \leq C^k\,.$$

Taking the root of this we find that $\|X\|_m := \mathbb{E}[|X|^m]^{\frac{1}{m}}$ is not dependent on m and scales as a constant, $\|X\|_m \asymp m^0$. ■

Proof of Theorem 4.1. We first consider the gradient for a single data-label pair of $W_{i,l,j}$ the weight that maps from neuron l in layer $i-1$ to neuron j in layer i. We study this

gradient using the chain rule, where we decompose $\partial E_{\mathcal{L}}(\cdot)/\partial W_{i,l,j}$ as

$$\frac{\partial E_{\mathcal{L}}(\cdot)}{\partial \widetilde{h}_{i,j}} \cdot \frac{\partial \widetilde{h}_{i,j}}{\partial W_{i,l,j}},$$

where $\widetilde{h}_{i,j}$ is the (noised) activation of the j^{th} neuron in the i^{th} layer. Thus, the gradient noise on the weights can be described as the product of two random variables.

Additive Noise. Let us first consider the properties of $\partial E_{\mathcal{L}}(\cdot)/\partial \widetilde{h}_{i,j}$ for the additive case.

Regression. In the case of regression we use a mean-square-error (MSE) we have that:

$$\Delta\mathcal{L}(\mathbf{x},\mathbf{y}) = 2(\mathbf{y} - \mathbf{h}_L(\mathbf{x}))\mathcal{E}_L(\mathbf{x}) + (\mathcal{E}_L(\mathbf{x}))^2 \ ,$$

where we imply all terms' dependence on $\mathbf{w}, \boldsymbol{\epsilon}$ for brevity of notation. One can already see that the derivative of this object with respect to each element of $\widetilde{\mathbf{h}}_L$ will have tail properties that are asymptotically equivalent to those of $\mathcal{E}_L$, which we know by Lemma 4.1.

$$\left\| \frac{\partial \Delta\mathcal{L}(\mathbf{x},\mathbf{y})}{\partial \widetilde{h}_{L,j}} \right\|_m \lesssim \sqrt{m}, \qquad m = 1, \dots, n_L \, .$$

If we center this distribution the tail properties of this variable are unchanged (Vershynin, 2018; Kuchibhotla and Chakrabortty, 2018). In particular the asymptotic behaviour of $\|\cdot\|_m$ is unchanged and we have that:

$$\begin{aligned} \left\| \frac{\partial \Delta\mathcal{L}(\mathbf{x},\mathbf{y})}{\partial \widetilde{h}_{L,j}} - \mathbb{E}_{\boldsymbol{\epsilon}}\left[\frac{\partial \Delta\mathcal{L}(\mathbf{x},\mathbf{y})}{\partial \widetilde{h}_{L,j}} \right] \right\|_m &\lesssim \sqrt{m}, \\ \Leftrightarrow \left\| \frac{\partial E_{\mathcal{L}}(\mathbf{x},\mathbf{y})}{\partial \widetilde{h}_{L,j}} \right\|_m &\lesssim \sqrt{m}, \qquad m = 1, \dots, n_L \, . \end{aligned}$$

Classification. In the case of classification we use a cross-entropy (CE) error. There is no easy closed-form for $\Delta\mathcal{L}(\mathbf{x},\mathbf{y})$ here, but we can infer the properties of $\nabla_{\widetilde{\mathbf{h}}_L}\Delta\mathcal{L}(\mathbf{x},\mathbf{y})$ by studying the properties of the gradient $\nabla_{\mathbf{h}_L}\mathcal{L}(\mathbf{x},\mathbf{y})$. For CE we know that:

$$\nabla_{\mathbf{h}_L}\mathcal{L}(\mathbf{x},\mathbf{y}) = \text{sigmoid}(\mathbf{h}_L(\mathbf{x})) - \mathbf{y}$$

in the binary label case. In the multi-label classification case we typically use a softmax, which is also a bounded function. We can already see that any noise $\mathcal{E}_L$ added to $\mathbf{h}_L(\mathbf{x})$ will induce a change in the gradient that is inherently bounded by the sigmoid non-linearity, meaning that $\Delta\mathcal{L}(\mathbf{x},\mathbf{y})$ will be bounded. As such the centered variable $E_{\mathcal{L}}(\mathbf{x},\mathbf{y})$ will also be bounded *and* zero mean,. By Lemma B.4 any bounded and zero mean distribution will be sub-Weibull with parameter $\theta = 0$, and will thus *also* be sub-Gaussian

$$\left\| \frac{\partial E_{\mathcal{L}}(\mathbf{x},\mathbf{y})}{\partial \widetilde{h}_{L,j}} \right\|_m \lesssim \sqrt{m}, \qquad m = 1,\dots,n_L\,.$$

Synthesizing the regression and classification settings we can conclude that each constitutive element of $\nabla_{\widetilde{\mathbf{h}}_L} E_{\mathcal{L}}(\mathbf{x},\mathbf{y})$ will be sub-Gaussian and will have zero mean.

We can now turn to the partial derivatives of the form $\partial E_{\mathcal{L}}(\cdot)/\partial \widetilde{h}_i^m$. Assume $\nabla_{\widetilde{\mathbf{h}}_i} E_{\mathcal{L}}(\mathbf{x},\mathbf{y})$ is of order m^r, with,

$$\left\| \frac{\partial E_{\mathcal{L}}(\mathbf{x},\mathbf{y})}{\partial \widetilde{h}_{i,j}} \right\|_m \lesssim m^r, \qquad m = 1,\dots,n_i\,,$$

which entails for gradients at the previous $(i-1)^{\text{th}}$ layer we have

$$\frac{\partial E_{\mathcal{L}}(\cdot)}{\partial \widetilde{h}_{i-1,l}} = \nabla_{\widetilde{\mathbf{h}}_i} E_{\mathcal{L}}(\mathbf{x},\mathbf{y}) \nabla_{\widetilde{h}_{i-1,l}} \widetilde{\mathbf{h}}_i(\mathbf{x})\,,$$

where

$$\nabla_{\widetilde{h}_{i-1,l}} \widetilde{\mathbf{h}}_i(\mathbf{x}) = \kappa'\left(\mathbf{W}_{i,l}\widetilde{h}_{i-1,l}(\mathbf{x})\right) \circ (\mathbf{W}_{i,l})\,,$$

where $\circ$ denotes the element wise product and $\mathbf{W}_{i,l}$ is the l^{th} column of the weight matrix $\mathbf{W}_i$. By definition, activation functions that obey the extended envelope property will have gradients that are bounded in norm, by some constant d_2. As such, by Lemma B.4 κ' will be sub-Weibull with $r = 0$. By Hölder's inequality we have that

$$\left\| \left(\nabla_{\widetilde{\mathbf{h}}_i} E_{\mathcal{L}}(\mathbf{x},\mathbf{y})\right)_z \left(\kappa'\left(\mathbf{W}_{i,l}\widetilde{h}_{i-1,l}(\mathbf{x})\right)\right)_j \right\|_m \leq \left\| \left(\nabla_{\widetilde{\mathbf{h}}_i} E_{\mathcal{L}}(\mathbf{x},\mathbf{y})\right)_j \right\|_{2m} \left\| \left(\kappa'\left(\mathbf{W}_{i,l}\widetilde{h}_{i-1,l}(\mathbf{x})\right)\right)_j \right\|_{2m},$$

where j indexes over the elements of both Jacobians. By definition, we know that there exists $A > 0$ and $B > 0$ such that $\|(\nabla_{\widetilde{\mathbf{h}}_i} E_{\mathcal{L}}(\mathbf{x},\mathbf{y}))_z\|_{2m} \leq A(2m)^r$ and $\|(\kappa'(\mathbf{W}_{i,l}\widetilde{h}_{i-1,l}(\mathbf{x})))_j\|_{2m} \leq$

B. As such

$$\left\|\left(\nabla_{\widetilde{\mathbf{h}}_i} E_{\mathcal{L}}(\mathbf{x},\mathbf{y})\right)_z \left(\kappa'\left(\mathbf{W}_{i,l}\widetilde{h}_{i-1,l}(\mathbf{x})\right)\right)_j\right\|_m \leq \left\|\left(\nabla_{\widetilde{\mathbf{h}}_i} E_{\mathcal{L}}(\mathbf{x},\mathbf{y})\right)_j\right\|_{2m} \left\|\left(\kappa'\left(\mathbf{W}_{i,l}\widetilde{h}_{i-1,l}(\mathbf{x})\right)\right)_j\right\|_{2m}$$
$$\leq AB2^r m^r\,.$$

Thus we know that the product of these two variables will be asymptotically upper-bounded by

$$\left\|\left(\nabla_{\widetilde{\mathbf{h}}_i} E_{\mathcal{L}}(\mathbf{x},\mathbf{y})\right)_j \left(\kappa'\left(\mathbf{W}_{i,l}\widetilde{h}_{i-1,l}(\mathbf{x})\right)\right)_j\right\|_m \lesssim m^r\,.$$

We now need to take into account the weighted sum across rows and columns (i.e. over indices j) that occurs. By Lemma B.3 we know that

$$\left\|\sum_{j=1}^{n_i}\left(\nabla_{\widetilde{\mathbf{h}}_i} E_{\mathcal{L}}(\mathbf{x},\mathbf{y})\right)_j \left(\kappa'\left(\mathbf{W}_{i,l}\widetilde{h}_{i-1,l}(\mathbf{x})\right)\circ(\mathbf{W}_{i,l})\right)_j\right\|_m \lesssim m^r,$$
$$\Leftrightarrow \left\|\frac{\partial E_{\mathcal{L}}(\cdot)}{\partial \widetilde{h}_{i-1,l}}\right\|_m \lesssim m^r, \qquad m=1,\ldots,n_i\,.$$

By recursion, with $\widetilde{\mathbf{h}}_L$ as the base case, gradients at layer i bounded in norm by m^r induce gradients at layer $i-1$, also bounded in norm by m^r. By recursion with the L^{th} layer as the base case we can claim that,

$$\left\|\frac{\partial E_{\mathcal{L}}(\cdot)}{\partial \widetilde{h}_{i,j}}\right\|_m \lesssim \sqrt{m}\,.$$

We have now defined the first constitutive term of $\partial E_{\mathcal{L}}(\cdot)/\partial W_{i,l,j}$. Defining $\partial \widetilde{h}_{i,j}/\partial W_{i,l,j}$ is much simpler:

$$\frac{\partial \widetilde{h}_{i,j}}{\partial W_{i,l,j}} = \kappa'\left(W_{i,l,j}\widetilde{h}_{i-1,l}(\mathbf{x})\right)\left(\widetilde{h}_{i-1,l}(\mathbf{x})\right)\,.$$

Here $\widetilde{h}_{i-1,l}(\mathbf{x})$, which we know is sub-Gaussian by Lemma 4.1, is once again multiplied to a bounded variable, κ'. Thus reapplying Hölder's inequality we obtain that

$$\left\|\frac{\partial \widetilde{h}_{i,j}}{\partial W_{i,l,j}}\right\|_m \lesssim \sqrt{m}\,.$$

We can now bring together the characterisations of the gradients that constitute

$\partial E_{\mathcal{L}}(\cdot)/\partial W_{i,l,j}$. We can re-use Hölder's inequality to show that the product of these variables will be sub-exponential

$$\left\| \frac{\partial E_{\mathcal{L}}((\mathbf{x},\mathbf{y});\mathbf{w},\boldsymbol{\epsilon})}{\partial W_{i,l,j}} \right\|_m \lesssim m, \qquad \text{for every } m \geq 1\,.$$

Multiplicative Noise. In the case of multiplicative noise we know that by Lemma 4.1, the accumulated noise at layer L will be of the same order as that at layer $L-1$, because we are not multiplying noise to the final layer, thus

$$\left\| \frac{\partial E_{\mathcal{L}}(\mathbf{x},\mathbf{y})}{\partial \widetilde{h}_{L,j}} \right\|_m \lesssim m^{\frac{L}{2}}, \qquad m = 1,\ldots,n_L\,.$$

Repeating the analysis done for the additive case we can claim that,

$$\left\| \frac{\partial E_{\mathcal{L}}(\cdot)}{\partial \widetilde{h}_i^m} \right\|_m \lesssim m^{\frac{L}{2}}.$$

We have now defined the first constitutive term of $\partial E_{\mathcal{L}}(\cdot)/\partial W_{i,l,j}$. We now define $\partial \widetilde{h}_{i,j}/\partial W_{i,l,j}$.

$$\frac{\partial \widetilde{h}_i^m}{\partial W_{i,l,j}} = \kappa' \left(W_{i,l,j} \widetilde{h}_{i-1,l}(\mathbf{x}) \right) \left(\widetilde{h}_{i-1,l}(\mathbf{x}) \right)\,.$$

Here $\widetilde{h}_{i-1,l}(\mathbf{x})$, which we know is sub-Weibull with a parameter $p = \frac{i}{2}$ by Lemma 4.1, is once again multiplied to a bounded variable, κ'. Thus reapplying Hölder's inequality we obtain that

$$\left\| \frac{\partial \widetilde{h}_i^m}{\partial W_{i,l,j}} \right\|_m \lesssim m^{\frac{i}{2}}.$$

We can re-use Hölder's inequality to show that the product of these variables will be sub-Weibull with $p = \frac{L+i}{2}$

$$\left\| \frac{\partial E_{\mathcal{L}}((\mathbf{x},\mathbf{y});\mathbf{w},\boldsymbol{\epsilon})}{\partial W_{i,l,j}} \right\|_m \lesssim m^{\frac{L+i}{2}}, \qquad \text{for every } m \geq 1\,.$$

Mean of Noise. Finally, by definition these gradients are zero mean,

$$\frac{\partial E_{\mathcal{L}}((\mathbf{x},\mathbf{y});\mathbf{w},\boldsymbol{\epsilon})}{\partial W_{i,l,j}} = \frac{\partial \Delta\mathcal{L}(\mathbf{x},\mathbf{y})}{\partial W_{i,l,j}} - \mathbb{E}_{\boldsymbol{\epsilon}}\left[\frac{\partial \Delta\mathcal{L}(\mathbf{x},\mathbf{y})}{\partial W_{i,l,j}} \right].$$

■

B.7.3 Proof of Theorem 4.5

Before we proceed to the proof of Theorem 4.5, we present some technical results that will be used in the proof of Theorem 4.5 later.

For the d-dimensional asymmetric fractional Langevin dynamics $\mathbf{w}_t$, its infinitesimal generator is given in the following proposition.

Proposition B.7.8. *The asymmetric fractional Langevin dynamics $\mathbf{w}_t$ has the infinitesimal generator:*

$$\mathcal{L}f(\mathbf{w}) = \sum_{i=1}^{d} \left((b(\mathbf{w}, \alpha, \theta))_i - \varepsilon^{\alpha} \frac{\theta_i}{\cos(\alpha\pi/2)} \frac{\alpha}{(\alpha-1)\Gamma(1-\alpha)} \right) \frac{\partial f(\mathbf{w})}{\partial w_i} + \varepsilon^{\alpha} \sum_{i=1}^{d} \mathcal{H}_{w_i}^{\alpha,\theta_i} f(\mathbf{w}), \tag{B.36}$$

where

$$\begin{aligned} \mathcal{H}_{w_i}^{\alpha,\theta_i} f(\mathbf{w}) := & \left(\frac{1}{2} + \frac{\theta_i}{2} \right) \frac{1}{\cos(\alpha\pi/2)} \frac{\alpha}{\Gamma(1-\alpha)} \int_0^{\infty} \frac{f(\mathbf{w} + \xi \mathbf{e}_i) - f(\mathbf{w}) - \partial_{w_i} f(\mathbf{w}) \xi}{\xi^{\alpha+1}} d\xi \\ & + \left(\frac{1}{2} - \frac{\theta_i}{2} \right) \frac{1}{\cos(\alpha\pi/2)} \frac{\alpha}{\Gamma(1-\alpha)} \int_0^{\infty} \frac{f(\mathbf{w} - \xi \mathbf{e}_i) - f(\mathbf{w}) + \partial_{w_i} f(\mathbf{w}) \xi}{\xi^{\alpha+1}} d\xi, \end{aligned} \tag{B.37}$$

where $\mathbf{e}_i$ is the i-th basis vector in $\mathbb{R}^d$, i.e. a d-dimensional unit vector with i-th coordinate being 1 and all the other coordinates being 0.

Proof of Proposition B.7.8. Since the asymmetric fractional Langevin dynamics is driven by the d-dimensional $\mathbf{L}_t^{\alpha,\theta}$, it suffices to show that the infinitesimal generator of d-dimensional $\mathbf{L}_t^{\alpha,\theta}$ is given by

$$\sum_{i=1}^{d} \mathcal{G}_{w_i}^{\alpha,\theta_i} f(\mathbf{w}) = \sum_{i=1}^{d} \mathcal{H}_{w_i}^{\alpha,\theta_i} f(\mathbf{w}) - \sum_{i=1}^{d} \frac{\theta_i}{\cos(\alpha\pi/2)} \frac{\alpha}{(\alpha-1)\Gamma(1-\alpha)} \frac{\partial}{\partial w_i} f(\mathbf{w}). \tag{B.38}$$

We start the proof by considering the dimension $d = 1$ first. The one-dimensional α-stable Lévy motion with tail-index $1 < \alpha < 2$ and skewness $\theta \in (-1, 1)$ has the infinitesimal

generator given by:

$$\mathcal{G}^{\alpha,\theta}f(w) := \frac{1+\theta}{2}\frac{1}{\cos(\alpha\pi/2)}\frac{\alpha}{\Gamma(1-\alpha)}\int_{z>0}[f(w+z)-f(w)-1_{|z|\leq 1}f'(w)z]\frac{dz}{|z|^{1+\alpha}} + \frac{1-\theta}{2}\frac{1}{\cos(\alpha\pi/2)}\frac{\alpha}{\Gamma(1-\alpha)}\int_{z<0}[f(w+z)-f(w)-1_{|z|\leq 1}f'(w)z]\frac{dz}{|z|^{1+\alpha}} + af'(w), \tag{B.39}$$

where $a \in \mathbb{R}$ is chosen so that $\mathcal{G}^{\alpha,\theta}w = 0$ to be consistent with $\mu = 0$ in $\mathbf{L}_t^{\alpha,\theta} - \mathbf{L}_s^{\alpha,\theta} \sim \mathcal{S}_\alpha((t-s)^{1/\alpha}, \theta, \mu)$ for any $t > s$. Thus, we can compute that

$$\mathcal{G}^{\alpha,\theta}w = \frac{1+\theta}{2}\frac{1}{\cos(\alpha\pi/2)}\frac{\alpha}{\Gamma(1-\alpha)}\int_{z>1}\frac{dz}{z^\alpha} - \frac{1-\theta}{2}\frac{1}{\cos(\alpha\pi/2)}\frac{\alpha}{\Gamma(1-\alpha)}\int_{z>1}\frac{dz}{z^\alpha} + a = 0, \tag{B.40}$$

which yields that

$$a = -\theta\frac{1}{\cos(\alpha\pi/2)}\frac{\alpha}{(\alpha-1)\Gamma(1-\alpha)}. \tag{B.41}$$

Therefore, with $1 < \alpha < 2$,

$$\mathcal{G}^{\alpha,\theta}f(w) = \mathcal{H}^{\alpha,\theta}f(w) - \theta\frac{1}{\cos(\alpha\pi/2)}\frac{\alpha}{(\alpha-1)\Gamma(1-\alpha)}f'(w), \tag{B.42}$$

where

$$\mathcal{H}^{\alpha,\theta}f(w) := \left(\frac{1}{2}+\frac{\theta}{2}\right)\frac{1}{\cos(\alpha\pi/2)}\frac{\alpha}{\Gamma(1-\alpha)}\int_0^\infty \frac{f(w+\xi)-f(w)-f'(w)\xi}{\xi^{\alpha+1}}d\xi + \left(\frac{1}{2}-\frac{\theta}{2}\right)\frac{1}{\cos(\alpha\pi/2)}\frac{\alpha}{\Gamma(1-\alpha)}\int_0^\infty \frac{f(w-\xi)-f(w)+f'(w)\xi}{\xi^{\alpha+1}}d\xi. \tag{B.43}$$

Similarly, for the multi-dimensional case, we can show that the infinitesimal generator for $\mathbf{L}_t^{\alpha,\theta}$ is given by:

$$\sum_{i=1}^d \mathcal{G}_{w_i}^{\alpha,\theta_i}f(\mathbf{w}) = \sum_{i=1}^d \mathcal{H}_{w_i}^{\alpha,\theta_i}f(\mathbf{w}) - \sum_{i=1}^d \frac{\theta_i}{\cos(\alpha\pi/2)}\frac{\alpha}{(\alpha-1)\Gamma(1-\alpha)}\frac{\partial}{\partial w_i}f(\mathbf{w}), \tag{B.44}$$

where

$$\mathcal{H}_{w_i}^{\alpha,\theta_i} f(\mathbf{w}) := \left(\frac{1}{2}+\frac{\theta_i}{2}\right)\frac{1}{\cos(\alpha\pi/2)}\frac{\alpha}{\Gamma(1-\alpha)}\int_0^\infty \frac{f(\mathbf{w}+\xi\mathbf{e}_i)-f(\mathbf{w})-\partial_{w_i}f(\mathbf{w})\xi}{\xi^{\alpha+1}}d\xi + \left(\frac{1}{2}-\frac{\theta_i}{2}\right)\frac{1}{\cos(\alpha\pi/2)}\frac{\alpha}{\Gamma(1-\alpha)}\int_0^\infty \frac{f(\mathbf{w}-\xi\mathbf{e}_i)-f(\mathbf{w})+\partial_{w_i}f(\mathbf{w})\xi}{\xi^{\alpha+1}}d\xi, \tag{B.45}$$

where $\mathbf{e}_i$ is the i-th basis vector in $\mathbb{R}^d$, i.e. a d-dimensional unit vector with i-th coordinate being 1 and all the other coordinates being 0. ■

We recall that for any $\theta_i \in (-1,1)$, $1 \le i \le d$, and $1 < \alpha < 2$, we have

$$(b(\mathbf{w},\alpha,\theta))_i = \frac{\varepsilon^\alpha}{\varphi(\mathbf{w})}\mathcal{D}_{w_i}^{\alpha-2,-\theta_i}(\partial_{w_i}\varphi(\mathbf{w})), \qquad \varphi(\mathbf{w}) = e^{-\varepsilon^{-\alpha}f(\mathbf{w})}, \tag{B.46}$$

where

$$\mathcal{D}_{w_i}^{\alpha-2,-\theta_i}(\partial_{w_i}\varphi(\mathbf{w})) := \frac{-1}{2\cos(\alpha\pi/2)}\left[(1-\theta_i)\mathcal{I}_{+,w_i}^{2-\alpha}(\partial_{w_i}\varphi(\mathbf{w})) + (1+\theta_i)\mathcal{I}_{-,w_i}^{2-\alpha}(\partial_{w_i}\varphi(\mathbf{w}))\right], \tag{B.47}$$

and

$$\mathcal{I}_{\pm,w_i}^{2-\alpha}(\partial_{w_i}\varphi(\mathbf{w})) := \frac{1}{\Gamma(2-\alpha)}\int_0^\infty \frac{\partial_{w_i}\varphi(\mathbf{w}\pm\xi\mathbf{e}_i)}{\xi^{\alpha-1}}d\xi. \tag{B.48}$$

In the next result, we provide an alternative formula for $b(\mathbf{w},\alpha,\theta) = ((b(\mathbf{w},\alpha,\theta))_i, 1 \le i \le d)$ that is defined in (4.19).

Proposition B.7.9. *For any $\theta_i \in (-1,1)$, $1 \le i \le d$, and $1 < \alpha < 2$, we have*

$$(b(\mathbf{w},\alpha,\theta))_i := \frac{\varepsilon^\alpha}{\varphi(\mathbf{w})}\left(\frac{1}{2}-\frac{\theta_i}{2}\right)\frac{1}{\cos(\alpha\pi/2)}\frac{\alpha}{\Gamma(1-\alpha)}\int_0^\infty \frac{\int_{w_i}^{w_i+\xi}\varphi(\mathbf{w}+(y-w_i)\mathbf{e}_i)dy - \varphi(\mathbf{w})\xi}{\xi^{\alpha+1}}d\xi + \frac{\varepsilon^\alpha}{\varphi(\mathbf{w})}\left(\frac{1}{2}+\frac{\theta_i}{2}\right)\frac{1}{\cos(\alpha\pi/2)}\frac{\alpha}{\Gamma(1-\alpha)}\int_0^\infty \frac{\int_{w_i}^{w_i-\xi}\varphi(\mathbf{w}+(y-w_i)\mathbf{e}_i)dy + \varphi(\mathbf{w})\xi}{\xi^{\alpha+1}}d\xi + \varepsilon^\alpha\theta_i\frac{1}{\cos(\alpha\pi/2)}\frac{\alpha}{(\alpha-1)\Gamma(1-\alpha)}, \tag{B.49}$$

with $\varphi(\mathbf{w}) := \exp(-\varepsilon^{-\alpha}f(\mathbf{w}))$.

Proof of Proposition B.7.9. Let us first consider the case when $\theta_i = 0$, $1 \le i \le d$. We have

$$\begin{aligned}\mathcal{D}_{w_i}^{\alpha-2}(\partial_{w_i}\varphi(\mathbf{w})) &:= -\mathcal{I}_{w_i}^{2-\alpha}(\partial_{w_i}\varphi(\mathbf{w})) \\ &= \frac{-1}{2\cos(\alpha\pi/2)}\left[\mathcal{I}_{+,w_i}^{2-\alpha}(\partial_{w_i}\varphi(\mathbf{w})) + \mathcal{I}_{-,w_i}^{2-\alpha}(\partial_{w_i}\varphi(\mathbf{w}))\right], \end{aligned} \tag{B.50}$$

where

$$\mathcal{I}_{+,w_i}^{2-\alpha}(\partial_{w_i}\varphi(\mathbf{w})) := \frac{1}{\Gamma(2-\alpha)}\int_0^\infty \frac{\partial_{w_i}\varphi(\mathbf{w}+\xi\mathbf{e}_i)}{\xi^{\alpha-1}}d\xi, \tag{B.51}$$

$$\mathcal{I}_{-,w_i}^{2-\alpha}(\partial_{w_i}\varphi(\mathbf{w})) := \frac{1}{\Gamma(2-\alpha)}\int_0^\infty \frac{\partial_{w_i}\varphi(\mathbf{w}-\xi\mathbf{e}_i)}{\xi^{\alpha-1}}d\xi. \tag{B.52}$$

Similarly, when $\theta_i \in (-1,1)$, $1 \le i \le d$, and $1 < \alpha < 2$, we have

$$(b(\mathbf{w},\alpha,\theta))_i = \frac{\varepsilon^\alpha}{\varphi(\mathbf{w})}\mathcal{D}_{w_i}^{\alpha-2,-\theta_i}(\partial_{w_i}\varphi(\mathbf{w})), \qquad \varphi(\mathbf{w}) = e^{-\varepsilon^{-\alpha}f(\mathbf{w})}, \tag{B.53}$$

where

$$\mathcal{D}_{w_i}^{\alpha-2,-\theta_i}(\partial_{w_i}\varphi(\mathbf{w})) := \frac{-1}{2\cos(\alpha\pi/2)}\left[(1-\theta_i)\mathcal{I}_{+,w_i}^{2-\alpha}(\partial_{w_i}\varphi(\mathbf{w})) + (1+\theta_i)\mathcal{I}_{-,w_i}^{2-\alpha}(\partial_{w_i}\varphi(\mathbf{w}))\right]. \tag{B.54}$$

■

Now, we are ready to prove Theorem 4.5.

Proof of Theorem 4.5. We recall from Proposition B.7.8 that the asymmetric fractional Langevin dynamics $\mathbf{w}_t$ has the infinitesimal generator:

$$\mathcal{L}f(\mathbf{w}) = \sum_{i=1}^d \left((b(\mathbf{w},\alpha,\theta))_i - \varepsilon^\alpha \frac{\theta_i}{\cos(\alpha\pi/2)}\frac{\alpha}{(\alpha-1)\Gamma(1-\alpha)}\right)\frac{\partial f(\mathbf{w})}{\partial w_i} + \varepsilon^\alpha \sum_{i=1}^d \mathcal{H}_{w_i}^{\alpha,\theta_i} f(\mathbf{w}), \tag{B.55}$$

where $\mathcal{H}_{w_i}^{\alpha,\theta_i} f(\mathbf{w})$ is given in (B.37).

It follows that the adjoint operator $\mathcal{L}^*$ of $\mathcal{L}$ is given by:

$$\mathcal{L}^* f(\mathbf{w}) = -\sum_{i=1}^{d} \frac{\partial}{\partial w_i} \left(\left((b(\mathbf{w}, \alpha, \theta))_i - \varepsilon^\alpha \frac{\theta_i}{\cos(\alpha\pi/2)} \frac{\alpha}{(\alpha-1)\Gamma(1-\alpha)} \right) f(\mathbf{w}) \right) + \varepsilon^\alpha \sum_{i=1}^{d} \mathcal{H}_{w_i}^{\alpha,-\theta_i} f(\mathbf{w}) \,. \tag{B.56}$$

The probability density function $p(\mathbf{w}, t)$ of the Lévy-driven SDE satisfies the Fokker-Planck equation Schertzer et al. (2001):

$$\begin{aligned} \partial_t p(\mathbf{w}, t) &= \mathcal{L}^* p(\mathbf{w}, t) \\ &= -\sum_{i=1}^{d} \frac{\partial}{\partial w_i} \left[\left((b(\mathbf{w}, \alpha, \theta))_i - \varepsilon^\alpha \theta_i \frac{1}{\cos(\alpha\pi/2)} \frac{\alpha}{(\alpha-1)\Gamma(1-\alpha)} \right) p(\mathbf{w}, t) \right] \\ &\quad + \varepsilon^\alpha \sum_{i=1}^{d} \mathcal{H}_{w_i}^{\alpha,-\theta_i} p(\mathbf{w}, t). \end{aligned} \tag{B.57}$$

We can compute that

$$\begin{aligned} &\sum_{i=1}^{d} \frac{\partial}{\partial w_i} \left[\left((b(\mathbf{w}, \alpha, \theta))_i - \varepsilon^\alpha \theta_i \frac{1}{\cos(\alpha\pi/2)} \frac{\alpha}{(\alpha-1)\Gamma(1-\alpha)} \right) \varphi(\mathbf{w}) \right] + \varepsilon^\alpha \sum_{i=1}^{d} \mathcal{H}_{w_i}^{\alpha,-\theta_i} \varphi(\mathbf{w}) \\ &= -\varepsilon^\alpha \sum_{i=1}^{d} \frac{\partial}{\partial w_i} \left[\left(\frac{1}{2} - \frac{\theta_i}{2} \right) \frac{1}{\cos(\alpha\pi/2)} \frac{\alpha}{\Gamma(1-\alpha)} \int_0^\infty \frac{\int_{w_i}^{w_i+\xi} \varphi(\mathbf{w} + (y - w_i)\mathbf{e}_i) dy - \varphi(\mathbf{w})\xi}{\xi^{\alpha+1}} d\xi \right] \\ &\quad - \varepsilon^\alpha \sum_{i=1}^{d} \frac{\partial}{\partial w_i} \left[\left(\frac{1}{2} + \frac{\theta_i}{2} \right) \frac{1}{\cos(\alpha\pi/2)} \frac{\alpha}{\Gamma(1-\alpha)} \int_0^\infty \frac{\int_{w_i-\xi}^{w_i} \varphi(\mathbf{w} + (y - w_i)\mathbf{e}_i) dy + \varphi(\mathbf{w})\xi}{\xi^{\alpha+1}} d\xi \right] \\ &\quad + \varepsilon^\alpha \sum_{i=1}^{d} \mathcal{H}_{w_i}^{\alpha,-\theta_i} \varphi(\mathbf{w}) \\ &= -\varepsilon^\alpha \sum_{i=1}^{d} \left(\frac{1}{2} - \frac{\theta_i}{2} \right) \frac{1}{\cos(\alpha\pi/2)} \frac{\alpha}{\Gamma(1-\alpha)} \int_0^\infty \frac{\varphi(\mathbf{w} + \xi\mathbf{e}_i) - \varphi(\mathbf{w}) - \partial_{w_i}\varphi(\mathbf{w})\xi}{\xi^{\alpha+1}} d\xi \\ &\quad - \varepsilon^\alpha \sum_{i=1}^{d} \left(\frac{1}{2} + \frac{\theta_i}{2} \right) \frac{1}{\cos(\alpha\pi/2)} \frac{\alpha}{\Gamma(1-\alpha)} \int_0^\infty \frac{\varphi(\mathbf{w} - \xi\mathbf{e}_i) - \varphi(\mathbf{w}) + \partial_{w_i}\varphi(\mathbf{w})\xi}{\xi^{\alpha+1}} d\xi \\ &\quad + \varepsilon^\alpha \sum_{i=1}^{d} \mathcal{H}_{w_i}^{\alpha,-\theta_i} \varphi(\mathbf{w}) = 0. \end{aligned}$$

Hence, we conclude that $\pi(d\mathbf{w}) = \exp(-\varepsilon^{-\alpha} f(\mathbf{w}))d\mathbf{w} / \int_{\mathbb{R}^d} \exp(-\varepsilon^{-\alpha}(\mathbf{w}))d\mathbf{w}$ is an invariant distribution of the asymmetric fractional Langevin dynamics (4.18). Finally, if $b(\mathbf{w}, \alpha, \theta)$ is Lipschitz continuous in $\mathbf{w}$, then $\pi(d\mathbf{w})$ is the unique invariant distribution of

(4.18), see e.g. Schertzer et al. (2001). ■

B.7.4 Proof of Theorem B.5.1

Theorem B.5.1 provides a first-order approximation of the fractional derivative $\mathcal{D}^{\gamma,-\theta}$ when $d=1$.

Based on the work of Meerschaert and Tadjeran (2004), we will show a first-order approximation for the asymmetric fractional derivative $\mathcal{D}^{-\gamma,-\theta}$ when $-1<\gamma<0$ by using the shifted Grünwald-Letnikov difference operators defined in (B.4) and (B.5). Before we proceed to the proof of Theorem B.5.1, we will first present the Fourier transform property from equations (1) and (12) in Tian et al. (2015).

Property B.7.1 (Tian et al. (2015))**.** *Let $-1<\gamma<0$ and $f\in L^1(\mathbb{R})$. The Fourier transform of $\mathcal{I}_-^{-\gamma}f$ and $\mathcal{I}_+^{-\gamma}f$ satisfy the following identities:*

$$\mathcal{F}\left[\mathcal{I}_-^{-\gamma}f(w)\right](\zeta)=(i\zeta)^{\gamma}\hat{f}(\zeta),\quad \mathcal{F}\left[\mathcal{I}_+^{-\gamma}f(w)\right](\zeta)=(-i\zeta)^{\gamma}\hat{f}(\zeta), \tag{B.58}$$

where $\hat{f}(\zeta)$ denotes the Fourier transform of f, such that $\hat{f}(\zeta)=\int_{-\infty}^{\infty}e^{-i\zeta w}f(w)dw$.

Now, we are ready to prove Theorem B.5.1.

Proof of Theorem B.5.1. The main idea for the proof of Theorem B.5.1 is to use the Fourier transform to estimate the difference between $\mathcal{F}\left[\left((1+\theta)\mathcal{A}_{h,p}^{\gamma}+(1-\theta)\mathcal{B}_{h,q}^{\gamma}\right)f(w)\right](\zeta)$ and $\mathcal{F}\left[\left((1+\theta)\mathcal{I}_-^{-\gamma}+(1-\theta)\mathcal{I}_+^{-\gamma}\right)f(w)\right](\zeta)$, and then apply the inverse Fourier transform to complete the proof. By the linearity of Fourier transforms, we can apply Fourier transform to (B.4) to obtain

$$\begin{aligned}\mathcal{F}\left[\mathcal{A}_{h,p}^{\gamma}f(w)\right](\zeta)&=\frac{1}{h^{\gamma}}\sum_{k=0}^{\infty}(-1)^k\binom{-\gamma+k-1}{k}e^{-i\zeta(k-p)h}\hat{f}(\zeta)\\&=\frac{1}{h^{\gamma}}e^{i\zeta ph}\left(1-e^{-i\zeta h}\right)^{\gamma}\hat{f}(\zeta)\\&=(i\zeta)^{\gamma}\left(\frac{1-e^{-i\zeta h}}{i\zeta h}\right)^{\gamma}e^{i\zeta ph}\hat{f}(\zeta)\\&=W_p(i\zeta h)(i\zeta)^{\gamma}\hat{f}(\zeta).\end{aligned} \tag{B.59}$$

Similarly, we can compute that

$$\begin{aligned}\mathcal{F}\left[\mathcal{B}^{\gamma}_{h,q}f(w)\right](\zeta) &= \frac{1}{h^{\gamma}}\sum_{k=0}^{\infty}(-1)^k\binom{-\gamma+k-1}{k}e^{i\zeta(k-q)h}\hat{f}(\zeta)\\ &= (-i\zeta)^{\gamma}\left(\frac{1-e^{i\zeta h}}{-i\zeta h}\right)^{\gamma}e^{-i\zeta qh}\hat{f}(\zeta)\\ &= W_{-q}(-i\zeta h)(-i\zeta)^{\gamma}\hat{f}(\zeta). \end{aligned} \tag{B.60}$$

In addition, since $W_p(z)$ and $W_{-q}(-z)$ are analytic for any complex number $|z|\leq 1$, there exist series expansions so that by the first-order Taylor expansion we have

$$\begin{aligned} W_p(z) &:= \left(\frac{1-e^{-z}}{z}\right)^{\gamma}e^{pz} = 1+\left(p-\frac{\gamma}{2}\right)z+\mathcal{O}\left(|z|^2\right),\\ W_{-q}(-z) &:= \left(\frac{1-e^{z}}{-z}\right)^{\gamma}e^{-qz} = 1-\left(q-\frac{\gamma}{2}\right)z+\mathcal{O}\left(|z|^2\right). \end{aligned} \tag{B.61}$$

Next, define a function $\hat{\psi}(h,\zeta)$ as the difference between $\mathcal{F}\left[\left((1+\theta)\mathcal{A}^{\gamma}_{h,p}+(1-\theta)\,\mathcal{B}^{\gamma}_{h,q}\right)f(w)\right](\zeta)$ and $\mathcal{F}\left[\left((1+\theta)\mathcal{I}^{-\gamma}_{-}+(1-\theta)\,\mathcal{I}^{-\gamma}_{+}\right)f(w)\right](\zeta)$. By the linearity of Fourier transform, we have

$$\begin{aligned}\hat{\psi}(h,\zeta) &= (1+\theta)\left(\mathcal{F}\left[\mathcal{A}^{\gamma}_{h,p}f(w)\right](\zeta)-\mathcal{F}\left[\mathcal{I}^{-\gamma}_{-}f(w)\right](\zeta)\right) \\ &\quad+(1-\theta)\left(\mathcal{F}\left[\mathcal{B}^{\gamma}_{h,q}f(w)\right](\zeta)-\mathcal{F}\left[\mathcal{I}^{-\gamma}_{+}f(w)\right](\zeta)\right) \\ &= (1+\theta)(i\zeta)^{\gamma}\hat{f}(\zeta)\left(W_p(i\zeta h)-1\right)+(1-\theta)(-i\zeta)^{\gamma}\hat{f}(\zeta)\left(W_{-q}(-i\zeta h)-1\right) \\ &= (1+\theta)(i\zeta)^{\gamma}\hat{f}(\zeta)\left(p-\frac{\gamma}{2}\right)(i\zeta h)-(1-\theta)(-i\zeta)^{\gamma}\hat{f}(\zeta)\left(q-\frac{\gamma}{2}\right)(i\zeta h) \\ &= (1+\theta)(i\zeta)^{\gamma+1}\hat{f}(\zeta)\left(p-\frac{\gamma}{2}\right)h+(1-\theta)(-i\zeta)^{\gamma+1}\hat{f}(\zeta)\left(q-\frac{\gamma}{2}\right)h \\ &\overset{(a)}{=} \left[(1+\theta)e^{i\pi\gamma/2}\left(p-\frac{\gamma}{2}\right)h-(1-\theta)e^{-i\pi\gamma/2}\left(q-\frac{\gamma}{2}\right)\right]|\zeta|^{1+\gamma}\hat{f}(\zeta)h \\ &\overset{(b)}{=} \Big[\cos\left(\frac{\gamma\pi}{2}\right)\left((p-q)+\theta(p+q-\gamma)\right) \\ &\qquad+\sin\left(\frac{\gamma\pi}{2}\right)\left((p+q-\gamma)+\theta(p-q)\right)i\Big]\,|\zeta|^{1+\gamma}\hat{f}(\zeta)h, \end{aligned} \tag{B.62, B.63}$$

where we used the fact that for any real x, $0<1+\gamma<1$, we have $ix=|x|e^{i\,\mathrm{sign}(x)\pi/2}$ so that $(ix)^{1+\gamma}=|x|^{1+\gamma}e^{i\,\mathrm{sign}(x)\pi(\gamma+1)/2}=\mathrm{sign}(x)|x|^{1+\gamma}e^{i\,\mathrm{sign}(x)\pi\gamma/2}$ which implies equality (a), and we applied Euler's formula with $-1<\gamma<0$ to get equality (b). By our assumption

$f \in \mathcal{C}^4(\mathbb{R})$, we have

$$|\hat{f}(\zeta)| \leq C(1+|\zeta|)^{-4},$$

for a constant $C > 0$ that may depend on f. Hence, by taking a sufficiently small h, we obtain,

$$\begin{aligned}
|\hat{\psi}(h,\zeta)| &\leq \Big|\Big[\cos^2\left(\frac{\gamma\pi}{2}\right)(p-q+\theta(p+q-\gamma))^2 \\
&\qquad + \sin^2\left(\frac{\gamma\pi}{2}\right)(p+q-\gamma+\theta(p-q))^2\Big]^{\frac{1}{2}} C(1+|\zeta|)^{\gamma-3}h + c_0h^2 \\
&\leq \Big[\cos\left(\frac{\gamma\pi}{2}\right)|p-q+\theta(p+q-\gamma)| \\
&\qquad + \Big|\sin\left(\frac{\gamma\pi}{2}\right)\Big||p+q-\gamma+\theta(p-q)|\Big] C(1+|\zeta|)^{\gamma-3}h + c_0h^2,
\end{aligned}$$

where we also used the inequality that $|\zeta|^{\gamma+1} \leq (1+|\zeta|)^{\gamma+1}$ for $-1 < \gamma < 0$, and $c_0 > 0$ is a constant may depend on p, q. When $f \in L^1(\mathbb{R})$, the inverse Fourier transform exists with $-1 < \gamma < 0$, i.e. $\psi(h,w) = \frac{1}{2\pi i}\int_{-\infty}^{\infty} e^{-i\zeta w}\hat{\psi}(h,\zeta)d\zeta$, and it follows that

$$\begin{aligned}
|\psi(h,w)| &= \frac{1}{2\pi}\int_{-\infty}^{\infty}\hat{\psi}(h,\zeta)e^{-i\zeta w}d\zeta \\
&\leq \frac{1}{2\pi}\int_{-\infty}^{\infty}|\hat{\psi}(h,\zeta)|d\zeta \\
&\leq \Big[\cos\left(\frac{\gamma\pi}{2}\right)|p-q+\theta(p+q-\gamma)| \\
&\qquad + + \Big|\sin\left(\frac{\gamma\pi}{2}\right)\Big||p+q-\gamma+\theta(p-q)|\Big]\ \frac{C}{2\pi(|\gamma|+2)}h + \mathcal{O}\left(h^2\right), \qquad \text{(B.64)}
\end{aligned}$$

where $\mathcal{O}(\cdot)$ hides the dependence on p, q and γ, and $C > 0$ is a constant that may depend on $f \in L^1(\mathbb{R}) \cap \mathcal{C}^4(\mathbb{R})$.

Hence, we conclude that

$$\begin{aligned}
&\left|\mathcal{D}^{\gamma,-\theta}f(w) - \Delta_{h,p,q}^{\gamma,-\theta}f(w)\right| \\
&= \frac{1}{2\cos(\pi\gamma/2)}\left|(1+\theta)\left(\mathcal{A}_{h,p}^{\gamma}f(w) - \mathcal{I}_{-}^{\gamma}f(w)\right) + (1-\theta)\left(\mathcal{B}_{h,q}^{\gamma}f(w) - \mathcal{I}_{+}^{\gamma}f(w)\right)\right| \\
&\leq \left[|p-q| + |\theta|(p+q-\gamma) + \left|\tan\left(\frac{\gamma\pi}{2}\right)\right|(p+q-\gamma+|\theta||p-q|)\right]\ \frac{C}{4\pi(|\gamma|+2)}h + \mathcal{O}\left(h^2\right).
\end{aligned} \qquad \text{(B.65)}$$

The proof is complete. ■

B.7.5 Proof of Corollary B.5.3

With the definitions of the truncated series $\mathcal{A}^{\gamma}_{h,p,K}$ defined in (B.10) and $\mathcal{B}^{\gamma}_{h,q,K}$ in (B.11), we are now ready to prove Corollary B.5.3.

Proof of Corollary B.5.3. We will first control the difference $\left|\Delta^{\gamma,-\theta}_{h,p,q}\partial_w\varphi(w) - \Delta^{\gamma,-\theta}_{h,p,q,K}\partial_w\varphi(w)\right|$. Then the triangular inequality can be applied with the fractional derivative approximation error bound in Theorem B.5.1 to get the numerical truncation error.

By using the definitions of $\mathcal{A}_{h,p}, \mathcal{B}_{h,q}$ and $\mathcal{A}_{h,p,K}, \mathcal{B}_{h,q,K}$, under the Assumption B.5.2, there exist two universal constants $C_p > 0$ and $C_q > 0$ so that

$$
\begin{aligned}
&\left|\Delta^{\gamma,-\theta}_{h,p,q}\partial_w\varphi(w) - \Delta^{\gamma,-\theta}_{h,p,q,K}\partial_w\varphi(w)\right| \\
&= \frac{1}{2|\cos(\pi\gamma/2)|}\Big|(1+\theta)\left(\mathcal{A}^{\gamma}_{h,p}\partial_w\varphi(w-(k-p)h) - \mathcal{A}^{\gamma}_{h,p,K}\partial_w\varphi(w-(k-p)h)\right) \\
&\qquad +(1-\theta)\left(\mathcal{B}^{\gamma}_{h,q}\partial_w\varphi(w+(k-q)h) - \mathcal{B}^{\gamma}_{h,q,K}\partial_w\varphi(w+(k-q)h)\right)\Big| \\
&\le \frac{1}{2|\cos(\pi\gamma/2)|}\frac{1}{\Gamma(-\gamma)}\frac{1}{h^{\gamma}}\left(\sum_{k=K+p+1}^{\infty}\frac{\Gamma(-\gamma+k)}{\Gamma(k+1)}(1+\theta)|\partial_w\varphi(w-(k-p)h)|\right. \\
&\qquad \left. + \sum_{k=K+q+1}^{\infty}\frac{\Gamma(-\gamma+k)}{\Gamma(k+1)}(1-\theta)|\partial_w\varphi(w+(k-q)h)|\right) \\
&\le \frac{(1+\theta)C_p}{h^{\gamma}}\sum_{k=K+p+1}^{\infty}\frac{\Gamma(-\gamma+k)}{\Gamma(k+1)}e^{-(k-p)h} + \frac{(1-\theta)C_q}{h^{\gamma}}\sum_{k=K+q+1}^{\infty}\frac{\Gamma(-\gamma+k)}{\Gamma(k+1)}e^{-(k-q)h}.
\end{aligned}
\tag{B.66}
$$

Next, by applying Stirling's formula, we have as $k \to \infty$:

$$
\begin{aligned}
\frac{\Gamma(-\gamma+k)}{\Gamma(k+1)} &\sim \frac{\sqrt{2\pi(k-1-\gamma)}\,(k-1-\gamma)^{k-1-\gamma}\,e^{-(k-1-\gamma)}}{\sqrt{2\pi k}\,k^k\,e^{-k}} \\
&= \frac{(k-1-\gamma)^{\,k-1/2-\gamma}}{k^{\,k+1/2}}e^{1+\gamma} \\
&= k^{-\gamma-1}\left(1-\frac{1+\gamma}{k}\right)^{k}\left(\frac{k-1-\gamma}{k}\right)^{-1/2-\gamma}e^{1+\gamma} \\
&\sim k^{-\gamma-1}.
\end{aligned}
$$

Therefore, it follows from (B.66) that

$$\begin{aligned}
&\left|\Delta_{h,p,q}^{\gamma,-\theta}\partial_w\varphi(w) - \Delta_{h,p,q,K}^{\gamma,-\theta}\partial_w\varphi(w)\right| \\
&\leq (1+\theta)C_p h \sum_{k=K+p+1}^{\infty}(hk)^{-\gamma-1}e^{-(k-p)h} + (1-\theta)C_q h \sum_{k=K+q+1}^{\infty}(hk)^{-\gamma-1}e^{-(k-p)h} \\
&\leq ((1+\theta)C_p + (1-\theta)C_q)\,\frac{1}{hK}, \qquad \text{(B.67)}
\end{aligned}$$

where we abused the notation such that C_p, C_q in (B.67) may differ from C_p, C_q in (B.66). Finally, the triangular inequality yields that

$$\begin{aligned}
&\left|\mathcal{D}^{\gamma,-\theta}\partial_w\varphi(w) - \Delta_{h,p,q,K}^{\gamma,-\theta}\partial_w\varphi(w)\right| \\
&\quad\leq \left|\mathcal{D}^{\gamma,-\theta}\partial_w\varphi(w) - \Delta_{h,p,q}^{\gamma,-\theta}\partial_w\varphi(w)\right| + \left|\Delta_{h,p,q}^{\gamma,-\theta}\partial_w\varphi(w) - \Delta_{h,p,q,K}^{\gamma,-\theta}\partial_w\varphi(w)\right| \\
&\quad\leq \left[|p-q| + |\theta|(p+q-\gamma) + \left|\tan\left(\frac{\gamma\pi}{2}\right)\right|(p+q-\gamma+|\theta||p-q|)\right]\frac{C}{4\pi(|\gamma|+2)}h \\
&\qquad\qquad + ((1+\theta)C_p + (1-\theta)C_q)\,\frac{1}{hK} + \mathcal{O}\left(h^2\right),
\end{aligned}$$

where C_p and C_q are two universal constants following Assumption B.5.2. The proof is completed. ■

B.7.6 Proof of Theorem 4.6

First, let us recall that $\tilde{\nu}_N(g) = \frac{1}{H_N}\sum_{k=1}^{N}\eta_k g(\mathbf{w}_k)$ is the sample average, where $\mathbf{w}_k$ satisfies the Euler-Maruyama discretisation with the approximated drift $b_{h,K}$:

$$\tilde{\mathbf{w}}_{n+1} = \tilde{\mathbf{w}}_n + \eta_{n+1}b_{h,K}(\tilde{\mathbf{w}}_n,\alpha,\theta) + \varepsilon\eta_{n+1}^{1/\alpha}\Delta\mathbf{L}_{n+1}^{\alpha,\theta}, \qquad \text{(B.68)}$$

The corresponding SDE of (B.68) is given as

$$d\tilde{\mathbf{w}}_t = b_{h,K}(\tilde{\mathbf{w}}_{t-},\alpha,\theta)dt + \varepsilon d\mathbf{L}_t^{\alpha,\theta}, \qquad \text{(B.69)}$$

and we define $\tilde{\nu}(g) = \int g(\mathbf{w})\tilde{\pi}(d\mathbf{w})$, where $\tilde{\pi}$ is the stationary distribution of (B.69).

Next, let us introduce the following assumption that is needed for Theorem 4.6.

Assumption B.7.10. *(i) Assume that the step sizes are decreasing and the sum diverges*

such that $\lim_{n\to\infty}\eta_n = 0\,, \lim_{N\to\infty} H_N = \infty$.

(ii) Let $V : \mathbb{R} \to \mathbb{R}_+^*$ *be a function in* $\mathcal{C}^2$*, if* $\lim_{|x|\to\infty} V(w) = \infty$*,* $|\partial_w V| \le C\sqrt{V}$ *with some constant* $C > 0$ *and* $\partial_x^2 V$ *is bounded. Then there exists* $a \in (0, 1]$*,* $\delta > 0$ *and* $\beta \in \mathbb{R}$*, such that* $|b|^2 \le CV^a$ *and* $b(\partial_w V) \le \beta - \delta V^a$ *with* b *defined in* (4.19)*. And the statement also holds for* $\tilde{b}$*.*

(iii) The SDEs defined in (4.18) *and* (B.69) *are geometrically ergodic with their unique invariant measures.*

Before we proceed to the proof of Theorem 4.6, let us state a technical lemma bounding the error of $|\mathbb{E}[g(\mathbf{w}_t)] - \mathbb{E}[g(\tilde{\mathbf{w}}_t)]|$, where $(\mathbf{w}_t)_{t\ge0}$ and $(\tilde{\mathbf{w}}_t)_{t\ge0}$ follow SDEs in (4.18) and (B.69).

Lemma B.5. *Let* $(\mathbf{w}_t)_{t\ge0}$ *and* $(\tilde{\mathbf{w}}_t)_{t\ge0}$ *follow SDEs in* (4.18) *and* (B.69) *and* g *be a given test function with bounded* $|\partial_w g|$*. Suppose* $K \in \mathbb{N}\cup\{0\}$ *is a constant satisfying Assumption B.5.2 with respect to* $\partial_x\varphi$ *holds then we have the following bound:*

$$\begin{aligned}|\mathbb{E}\left[g\left(\mathbf{w}_t\right)\right] - \mathbb{E}\left[g\left(\tilde{\mathbf{w}}_t\right)\right]| \le \frac{\tilde{C}}{4\pi(|\gamma|+2)}&\left[|p-q| + |\theta|(p+q-\gamma)\right.\\ &\left.+\left|\tan\left(\frac{\gamma\pi}{2}\right)\right|(p+q-\gamma+|\theta||p-q|)\right]h\\ &+\left((1+\theta)C_p' + (1-\theta)C_q'\right)\frac{1}{hK} + \mathcal{O}\left(h^2\right)\,, \quad \text{(B.70)}\end{aligned}$$

where the constants $\tilde{C}, C_p', C_q' > 0$ *may depend on the function* $\partial_w\varphi$ *and the bound for* $|\partial_w g|$*.*

Proof. The proof is inspired by the proof of Lemma 3 in Şimşekli (2017). Let $\{P_t^{\mathbf{w}}\}_{t\ge0}$ and $\{P_t^{\tilde{\mathbf{w}}}\}_{t\ge0}$ be the corresponding Markov semigroups, i.e. $P_t^{\mathbf{w}} g(w) = \mathbb{E}_w[g(\mathbf{w}_t)]$, $P_t^{\tilde{\mathbf{w}}} g(w) = \mathbb{E}_w[g(\tilde{\mathbf{w}}_t)]$. Using the Markov semigroup property, following Lemma 3 in Şimşekli (2017), we have

$$|\mathbb{E}\left[g\left(\mathbf{w}_t\right)\right] - \mathbb{E}[g(\tilde{\mathbf{w}}_t)]| = \left|\int_0^t P_s^{\mathbf{w}}\left(\mathcal{L}^{\mathbf{w}} - \mathcal{L}^{\tilde{\mathbf{w}}}\right)P_{t-s}^{\tilde{\mathbf{w}}} g(w)ds\right|\,,$$

where $\mathcal{L}^{\mathbf{w}}$ and $\mathcal{L}^{\tilde{\mathbf{w}}}$ are the linear generators of $P_t^{\mathbf{w}}$ and $P_t^{\tilde{\mathbf{w}}}$, such that, for $g \in L^2(\pi)$, $\partial_t P_t g = \mathcal{L}P_t g = P_t\mathcal{L}g$. The infinitesimal generators $\mathcal{L}^{\mathbf{w}}$ and $\mathcal{L}^{\tilde{\mathbf{w}}}$ are computed in (B.55). By the

interchangeability of integration and differentiation, we have

$$\left|\int_0^t P_s^{\mathbf{w}}\left(\mathcal{L}^{\mathbf{w}}-\mathcal{L}^{\tilde{\mathbf{w}}}\right)P_{t-s}^{\tilde{\mathbf{w}}}g(w)ds\right| = \left|\int_0^t P_s^{\mathbf{w}}\left(b(w,\alpha,\theta)-b_{h,K}(w,\alpha,\theta)\right)P_{t-s}^{\tilde{\mathbf{w}}}\partial_w g(w)ds\right|.$$

By the ergodicity assumptions, for a bounded function f, there exist some constants $c>0$ and $\lambda_w,\lambda_{\tilde{w}}>0$ so that

$$|P_s^{\mathbf{w}}f| \le c\,e^{-\lambda_w s}\|f\|_\infty, \qquad |P_{t-s}^{\tilde{\mathbf{w}}}f| \le c\,e^{-\lambda_{\tilde{w}}(t-s)}\|f\|_\infty. \tag{B.71}$$

Using the boundedness assumption for $|\partial_w g|$ and Corollary B.5.3, and the fact that $\int_0^t e^{-\lambda_w s}ds \le \frac{1}{\lambda_w}$, we have

$$\begin{aligned}
&|\mathbb{E}[g(\mathbf{w}_t)]-\mathbb{E}[g(\tilde{\mathbf{w}}_t)]| \\
&\le \frac{\tilde{C}}{4\pi(|\gamma|+2)}\left[|p-q|+|\theta|(p+q-\gamma)+\left|\tan\left(\frac{\gamma\pi}{2}\right)\right|(p+q-\gamma+|\theta||p-q|)\right]h \\
&\qquad + \left((1+\theta)C_p'+(1-\theta)C_q'\right)\frac{1}{hK}+\mathcal{O}\left(h^2\right),
\end{aligned} \tag{B.72}$$

where the constants $\tilde{C}=\frac{c}{\lambda_w}C$, $C_p'=\frac{c}{\lambda_w}C_p$ and $C_q'=\frac{c}{\lambda_w}C_q$ may depend on $\partial_w\varphi$ and the bound for $|\partial_w g|$. ■

Now we are ready to prove Theorem 4.6.

Proof of Theorem 4.6. With the ergodicity assumptions, we have

$$\left|\nu(g)-\lim_{N\to\infty}\tilde{\nu}_N(g)\right| = \left|\nu(g)-\tilde{\nu}(g)+\tilde{\nu}(g)-\lim_{N\to\infty}\tilde{\nu}_N(g)\right| \tag{B.73}$$

$$\le \lim_{t\to\infty}\left|\mathbb{E}\left[g(\mathbf{w}_t)\right]-\mathbb{E}\left[g(\tilde{\mathbf{w}}_t)\right]\right| + \left|\tilde{\nu}(g)-\lim_{N\to\infty}\tilde{\nu}_N(g)\right|. \tag{B.74}$$

By Assumption B.7.10, Panloup (2008) and similar arguments as in Şimşekli (2017), we get,

$$\left|\tilde{\nu}(g)-\lim_{N\to\infty}\tilde{\nu}_N(g)\right| = 0, \quad \text{a.s.}$$

By applying Lemma B.5 as $t \to \infty$, we obtain:

$$\begin{aligned}\left|\nu(g) - \lim_{N\to\infty} \tilde{\nu}_N(g)\right| &\leq \lim_{t\to\infty} |\mathbb{E}[g(\mathbf{w}_t)] - \mathbb{E}[g(\tilde{\mathbf{w}}_t)]| \\ &\leq \frac{\tilde{C}}{4\pi(|\gamma|+2)}\left[|p-q| + |\theta|(p+q-\gamma) + \left|\tan\left(\frac{\gamma\pi}{2}\right)\right|(p+q-\gamma+|\theta||p-q|)\right] h \\ &\quad + \left((1+\theta)C'_p + (1-\theta)C'_q\right)\frac{1}{hK} + \mathcal{O}\left(h^2\right), \qquad \text{(B.75)}\end{aligned}$$

where $\tilde{C}$, C'_p, $C'_q > 0$ are constants that may depend on $\partial_w\varphi$ and the bound of $|\partial_w g|$. Finally, by taking $p = q = 0$, we complete the proof. ■

APPENDIX C

APPENDIX: TOWARDS A THEORETICAL UNDERSTANDING OF THE ROBUSTNESS OF VARIATIONAL AUTOENCODERS

C.1 Choosing r for r-robustness . . . 160

C.2 Proof of Theorem 5.1 . . . 163

C.3 Proof of Theorem 5.2 . . . 166

C.4 Empirical Calculation of the Bounds . . . 168

C.4.1 Estimating the minimum r . . . 168

C.4.2 Results . . . 168

C.4.3 Algorithm . . . 168

C.5 Estimating the Margin of Robustness . . . 169

C.6 -VAE Sensitivity Experiments . . . 170

C.7 Network Hyperparameters . . . 173

C.1 Choosing r for r-robustness

Proposition C.1.1. *For any input and perturbation, a necessary requirement for a VAE with a Gaussian encoder to satisfy r-robustness is that*

$$r > \sqrt{2\mathrm{Tr}(\boldsymbol{\Sigma}(\mathbf{x}))} + \mathcal{O}(\varepsilon) \tag{C.1}$$

where $\mathbf{\Sigma}(\mathbf{x}) = \mathbf{J}_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))\boldsymbol{\sigma}^2_\phi(\mathbf{x})\mathbf{J}^T_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))$, $(\mathbf{J}_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))_{i,j} = \partial g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))_i/\partial(\boldsymbol{\mu}_\phi(\mathbf{x}))_j$, *and* $\mathcal{O}(\boldsymbol{\varepsilon})$ *represents higher order terms that tend to zero in the limit* $\boldsymbol{\sigma}_\phi(\mathbf{x}) \to \mathbf{0}$.

We provide empirical confirmations in Appendix C.4.2 that show that the r for r-robustness scales with the encoder variance.

Proof. Let $g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))$ be the result of mapping to the encoder mean ($\boldsymbol{\mu}_\phi$) and then decoding to the likelihood mean (g_θ), and $\Delta(\mathbf{x}) = g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\eta} \circ \boldsymbol{\sigma}_\phi(\mathbf{x})) - g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))$ we want to find a bound for r for which:

$$p(||\Delta(\mathbf{x})||_2 \leq r) > p(||\Delta(\mathbf{x})||_2 > r) \tag{C.2}$$

where as before $\boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Here we can invoke Taylor's theorem on $g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\eta} \circ \boldsymbol{\sigma}_\phi(\mathbf{x}))$ around the deterministic mapping $\boldsymbol{\mu}_\phi(\mathbf{x})$. Namely, if we assume that all terms in Hessian of $g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x})$ are finite, i.e.

$$|\partial^2 g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x})_i/\partial \boldsymbol{\mu}_\phi(\mathbf{x})_j \boldsymbol{\mu}_\phi(\mathbf{x})_k| < \infty \, \forall i, j, k$$

, then we have:

$$g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\epsilon}) = g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x})) + \mathbf{J}_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))(\boldsymbol{\eta} \circ \boldsymbol{\sigma}_\phi(\mathbf{x})) + \mathcal{O}(\boldsymbol{\varepsilon}) \tag{C.3}$$

where $\mathcal{O}(\varepsilon)$ represents asymptotically dominated higher order terms that go to zero in the limit of small $\boldsymbol{\sigma}_\phi(\mathbf{x}))$ and $\mathbf{J}_\theta$ is defined element-wise as:

$$\mathbf{J}_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))_{i,j} = \frac{\partial g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))_i}{\partial(\boldsymbol{\mu}_\phi(\mathbf{x}))_j} \tag{C.4}$$

Note that $\mathbf{J}_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))(\boldsymbol{\eta} \circ \boldsymbol{\sigma}_\phi(\mathbf{x}))$ is distributed according to the multivariate Gaussian

$$\mathcal{N}(0, \mathbf{J}_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))\boldsymbol{\sigma}^2_\phi(\mathbf{x})\mathbf{J}^T_\theta(\boldsymbol{\mu}_\phi(\mathbf{x})))$$

Given these definitions

$$p(||\Delta(\mathbf{x})||_2 \leq r) > p(||\Delta(\mathbf{x})||_2 > r) \tag{C.5}$$

$$\Leftrightarrow p(||\Delta(\mathbf{x})||_2 \leq r) > 0.5 \tag{C.6}$$

$$\Leftrightarrow p(||\mathbf{J}_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))(\boldsymbol{\eta} \circ \boldsymbol{\sigma}_\phi(\mathbf{x})) + \mathcal{O}(\boldsymbol{\varepsilon})||_2 < r) > 0.5 \tag{C.7}$$

$$\Leftrightarrow p(||\mathbf{J}_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))(\boldsymbol{\eta} \circ \boldsymbol{\sigma}_\phi(\mathbf{x})) + \mathcal{O}(\boldsymbol{\varepsilon})||_2^2 < r^2) > 0.5 \tag{C.8}$$

We must now consider the distribution of the square norm of $\boldsymbol{\mathcal{E}}(\mathbf{x}) = \mathbf{J}_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))(\boldsymbol{\eta} \circ \boldsymbol{\sigma}_\phi(\mathbf{x}))$. Let

$$Q(\boldsymbol{\mathcal{E}}(\mathbf{x})) = ||\boldsymbol{\mathcal{E}}(\mathbf{x})||_2^2 = \boldsymbol{\mathcal{E}}(\mathbf{x})^T \boldsymbol{\mathcal{E}}(\mathbf{x}) \tag{C.9}$$

$$\boldsymbol{\Sigma}(\mathbf{x}) = \mathbf{J}_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))\boldsymbol{\sigma}_\phi^2(\mathbf{x})\mathbf{J}_\theta^T(\boldsymbol{\mu}_\phi(\mathbf{x})) \tag{C.10}$$

$$\mathbf{Y}(\mathbf{x}) = \boldsymbol{\Sigma}(\mathbf{x})^{-\frac{1}{2}} \boldsymbol{\mathcal{E}}(\mathbf{x}) \tag{C.11}$$

Given that we restrict ourselves to positive activation functions, $\mathbf{J}_\theta$ is positive and $\boldsymbol{\Sigma}(\mathbf{x})$ will be positive semi definite and is invertible. As such we have $Q(\boldsymbol{\mathcal{E}}) = \mathbf{Y}^T(\mathbf{x})\boldsymbol{\Sigma}(\mathbf{x})\mathbf{Y}(\mathbf{x})$.

Using the spectral decomposition theorem we can write that $\boldsymbol{\Sigma}(\mathbf{x}) = \mathbf{P}^T(\mathbf{x})\boldsymbol{\Lambda}(\mathbf{x})\mathbf{P}(\mathbf{x})$ where $\mathbf{P}^T(\mathbf{x})\mathbf{P}(\mathbf{x}) = \mathbf{I}$ and $\boldsymbol{\Lambda}(\mathbf{x})$ is the diagonal matrix of the eigenvalues of $\boldsymbol{\Sigma}(\mathbf{x})$, $\lambda_1, \ldots, \lambda_{d_\mathcal{X}}$, where $d_\mathcal{X}$ is the dimensionality of the data-space. Given that $\boldsymbol{\Sigma}(\mathbf{x})$ is positive semi definite $\boldsymbol{\Lambda}(\mathbf{x})$ will only have positive values.

Let $\mathbf{U}(\mathbf{x}) = \mathbf{P}(\mathbf{x})\mathbf{Y}(\mathbf{x}) = \mathbf{P}(\mathbf{x})\boldsymbol{\Sigma}(\mathbf{x})^{-\frac{1}{2}}\boldsymbol{\mathcal{E}}(\mathbf{x})$, which is multivariate Gaussian with identity matrix and zero mean. We have that:

$$Q(\boldsymbol{\mathcal{E}}) = \mathbf{Y}^T(\mathbf{x})\boldsymbol{\Sigma}(\mathbf{x})\mathbf{Y}(\mathbf{x}) \tag{C.12}$$

$$= \mathbf{Y}^T(\mathbf{x})\mathbf{P}^T(\mathbf{x})\boldsymbol{\Lambda}(\mathbf{x})\mathbf{P}(\mathbf{x})\mathbf{Y}(\mathbf{x}) \tag{C.13}$$

$$= \mathbf{U}^T(\mathbf{x})\boldsymbol{\Lambda}(\mathbf{x})\mathbf{U}(\mathbf{x}) \tag{C.14}$$

As such:

$$\sum_{i=1}^{d_{\mathcal{X}}}(\boldsymbol{\mathcal{E}}_i)^2 = \mathbf{U}^T(\mathbf{x})\boldsymbol{\Lambda}(\mathbf{x})\mathbf{U}(\mathbf{x}) = \sum_{i=1}^{d_{\mathcal{X}}}\lambda_i(\mathbf{U}_i(\mathbf{x}))^2, \quad \lambda_i(\mathbf{U}_i(\mathbf{x}))^2 \sim \Gamma\left(\frac{1}{2}, 2\lambda_i\right) \tag{C.15}$$

This comes from the fact that for $\lambda_i\mathbf{X}, \mathbf{X} \sim \Gamma\left(\frac{1}{2}, 2\right)$ we have that $\lambda_i\mathbf{X} \sim \Gamma\left(\frac{1}{2}, 2\lambda_i\right)$.

To establish a lower bound on r, we use Markov's inequality which states that:

$$p(||\boldsymbol{\mathcal{E}}(\mathbf{x}) + \mathcal{O}(\boldsymbol{\varepsilon})||_2^2 > r^2) < \frac{\mathbb{E}\,||\boldsymbol{\mathcal{E}}(\mathbf{x}) + \mathcal{O}(\boldsymbol{\varepsilon})||_2^2}{r^2} \tag{C.16}$$

Here $\mathbb{E}\,||\boldsymbol{\mathcal{E}}(\mathbf{x})||_2^2 = \mathbb{E}\sum_{i=1}^{d_{\mathcal{X}}}(\boldsymbol{\mathcal{E}}_i(\mathbf{x}))^2 = \mathbb{E}\sum_{i=1}^{d_{\mathcal{X}}}(\lambda_i(\mathbf{U}_i)^2(\mathbf{x}))^2$, which is simply $\sum_{i=1}^{d_{\mathcal{X}}}\lambda_i$.

Recall that we want: $p(||\boldsymbol{\mathcal{E}}(\mathbf{x}) + \mathcal{O}(\boldsymbol{\varepsilon})||_2^2 > r^2) < 0.5$. As such

$$r > \sqrt{2\sum_{i=1}^{d_{\mathcal{X}}}\lambda_i + \mathcal{O}(\boldsymbol{\varepsilon})} = \sqrt{2\mathrm{Tr}(\boldsymbol{\Sigma}(\mathbf{x}))} + \mathcal{O}(\boldsymbol{\varepsilon}) \tag{C.17}$$

■

C.2 Proof of Theorem 5.1

Proof. Suppose we have an r for which r-robustness is satisfied before any perturbation is added to the VAE input. First we want to establish a margin in the latent space $\mathcal{Z}$ for which our model is robust given a perturbation in the latent space.

To do this, we first define

$$\Delta_e(\mathbf{y}) = g_\theta(\mathbf{y}) - g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x})), \quad \mathbf{y} \in \mathcal{Z} \tag{C.18}$$

where g_θ is the decoder network and $\mathbf{y}$ is an arbitrary realization of the latents. Note here that there is an implicit dependency on $\mathbf{x}$, but as this input is fixed we will ignore this dependency throughout. Let $\mathbf{z} = \boldsymbol{\mu}_\phi(\mathbf{x}) + \boldsymbol{\eta} \circ \boldsymbol{\sigma}_\phi(\mathbf{x})$ be the random variable produced by the embedding, i.e. the latent sampled by the encoder. We want to find a bound R_e^r for

which:

$$\|\boldsymbol{\delta}_z\|_2 \leq R_e^r \quad \Leftrightarrow \quad p(||\Delta_e(\mathbf{z}+\boldsymbol{\delta}_z)||_2 \leq r) > p(||\Delta_e(\mathbf{z}+\boldsymbol{\delta}_z)||_2 > r) \tag{C.19}$$

such that r-robustness is satisfied on the decoder output when we apply a deterministic a perturbation $\boldsymbol{\delta}_z$ of maximum size R_e^r to the random variable $\mathbf{z}$. Note that all the stochasticity is contained in $\boldsymbol{\eta}$.

Let A^r denote the set of $\boldsymbol{\delta}_z$ for which (C.19) holds and conversely let B^r be the set of $\boldsymbol{\delta}_z$ for which it does not. By assumption in the Theorem, then $\mathbf{0} \in A^r$ as the unperturbed input satisfies r-robustness. Moreover, we also have that this unperturbed input $\boldsymbol{\delta}_z$ has a probability $p_\Delta(\mathbf{0}) := p(||\Delta_e(\mathbf{z})||_2 \leq r) = p(||\Delta(\mathbf{x})||_2 \leq r) > 0.5$ of returning a reconstruction with r of $g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))$.

Now we know that $\mathbf{z}$ is a Gaussian random variable and so regardless of form of the decoder, $p_\Delta(\boldsymbol{\delta}_z) := p(||\Delta_e(\mathbf{z}+\boldsymbol{\delta}_z)||_2 \leq r)$ must vary smoothly as we change $\boldsymbol{\delta}_z$. In essence, as we increase the size of the perturbation $\boldsymbol{\delta}_z$ slowly from zero, the distribution of $\mathbf{z}+\boldsymbol{\delta}_z$ will still have most of its mass of the same region $\mathbf{z}$. When coupled with the fact that we have some "excess probability" $p_\Delta(\mathbf{0})-0.5$ beyond what it is needed for r-robustness, there must be at certain degree to which we can increase $\boldsymbol{\delta}_z$ before all this excess probability is "used up". We can then use this to construct a bound for R_e^r by considering the minimum $\boldsymbol{\delta}_z$ to break r-robustness in the "worst-case" setting for the boundary between A^r and B^r.

Intuitively as shown in Figure C.2.1, and also more formally using the Neyman-Pearson lemma (Neyman and Pearson, 1933) by analogy to the approach of Cohen et al. (2019), this worst case setting will occur when the boundary between A^r and B^r is a straight line perpendicular to the direction of lowest variance for $\mathbf{z}$ (remembering that this is Gaussian distributed) and $\boldsymbol{\delta}_z$ is increased in this direction of lowest variance. In essence, this is the setup where our excess probability is used up most quickly for a given $\|\boldsymbol{\delta}_z\|_2$. By assumption in the theorem statement, we are using a diagonal covariance encoder and so this direction of lowest variance is the latent variable corresponding to $\arg\min_i \boldsymbol{\sigma}_\phi(\mathbf{x})_i$. Further, by noting that we need only consider the marginal distribution in this dimension,

it is straightforward to see that the bound is reached when

$$\|\boldsymbol{\delta}_z\|_2 = \left(\min_i \boldsymbol{\sigma}_\phi(\mathbf{x})_i\right) \Phi^{-1}\left(p_\Delta(\mathbf{0})\right) = \left(\min_i \boldsymbol{\sigma}_\phi(\mathbf{x})_i\right) \Phi^{-1}\left(p(||\Delta(\mathbf{x})||_2 \leq r)\right) \tag{C.20}$$

where Φ^{-1} is the inverse cumulative distribution function for a unit Gaussian, i.e. the probit function. Note that this yields $\|\boldsymbol{\delta}_z\|_2 = 0$ if $p(||\Delta(\mathbf{x})||_2 \leq r) = 0$, such we get the excepted result that our margin is zero is r-robustness only just holds without an input perturbation.

Next we need to relate $\|\boldsymbol{\delta}_z\|_2$ to $\|\boldsymbol{\delta}_x\|_2$. Here we can straightforwardly invoke Taylor's theorem on $\boldsymbol{\mu}_\phi(\mathbf{x} + \boldsymbol{\delta}_x)$ around the original input $\mathbf{x}$. Namely, if we assume that all terms in Hessian of $\mu_\phi(\mathbf{x})$ are finite (i.e. $|\partial^2 \boldsymbol{\mu}_\phi(\mathbf{x})_i / \partial \mathbf{x}_j \mathbf{x}_k| < \infty \, \forall i, j, k$), then we have

$$\boldsymbol{\delta}_z = \boldsymbol{\mu}_\phi(\mathbf{x} + \boldsymbol{\delta}_x) - \boldsymbol{\mu}_\phi(\mathbf{x}) = \mathbf{J}^\mu_\phi(\mathbf{x})\boldsymbol{\delta}_x + \mathcal{O}(\varepsilon) \tag{C.21}$$

where $\mathcal{O}(\varepsilon)$ represents asymptotically dominated higher order terms that go to zero in the limit of small $\boldsymbol{\delta}_x$. We thus have

$$\|\boldsymbol{\delta}_x\|_2 \leq \frac{\|\boldsymbol{\delta}_z\|_2}{\|\mathbf{J}^\mu_\phi(\mathbf{x})\|_F} + \mathcal{O}(\varepsilon) \tag{C.22}$$

where $\mathcal{O}(\varepsilon)$ again represents asymptotically dominated higher order terms (note though these are not the same terms as in (C.21)). To complete the proof we now simply combine this with (C.20) to give the $\|\boldsymbol{\delta}_x\|_2$ at which the bound is reached and thus the $R^r_{\mathcal{X}}(\mathbf{x})$ quoted in the theorem, namely

$$R^r_{\mathcal{X}}(\mathbf{x}) \geq \frac{(\min_i \boldsymbol{\sigma}_\phi(\mathbf{x})_i)\Phi^{-1}(p(||\Delta(\mathbf{x})||_2 \leq r))}{||\mathbf{J}^\mu_\phi(\mathbf{x})||_F} + \mathcal{O}(\varepsilon) \tag{C.23}$$

where the inequality comes from the fact that the $\boldsymbol{\delta}_z$ we derived was the worst possible case (i.e. smallest $\boldsymbol{\delta}_z$ which might reach the bound).

■

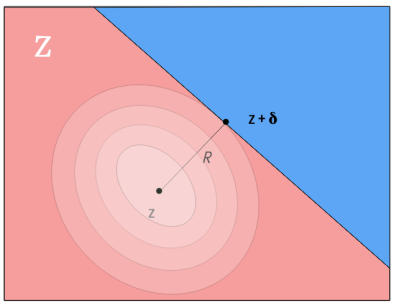


Figure C.2.1: Illustration of the boundary R we are measuring in $\mathcal{Z}$. Red represents spaces where A^r is satisfied. Blue represent spaces where B^r is satisfied. The concentric ellipsoids centered on $\mathbf{z}$ are the contours of $\mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{x}), \boldsymbol{\sigma}^2_\phi(\mathbf{x}))$. R is the minimum distance δ for which A^r is satisfied. The line dividing the two spaces represent the Neyman-Pearson "worst-case" model and is along the direction of minimum *variance*, $\min_i \boldsymbol{\sigma}^2_\phi(\mathbf{x})_i$.

C.3 Proof of Theorem 5.2

Proof. Here we use calculus of variations to obtain optimal posteriors for β-VAEs. The objectives we are optimising are over the whole dataset $\mathcal{D} = \{\mathbf{x}_i\}, i = 1, .., N$, with empirical data density $\rho(\mathbf{x}) = \frac{1}{N}\sum_i^N \delta(\mathbf{x} - \mathbf{x}_i)$.

The evidence lower bound for a β-VAE is

$$\mathcal{L}_\beta(\mathcal{D}; \theta, \phi) = \mathbb{E}_{\rho(\mathbf{x})}\left[\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - \beta\,\mathrm{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))\right]. \tag{C.24}$$

This is easier to work with written explicitly as integrals. Note that as we are going to be finding the optimal $q_\phi(\mathbf{z}|\mathbf{x})$ we must add a constraint so that it integrates to 1.

$$\mathcal{L}_\beta(\mathcal{D}; \theta, \phi) = \int \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{z}\rho(\mathbf{x})\left[q_\phi(\mathbf{z}|\mathbf{x})[\log p_\theta(\mathbf{x}|\mathbf{z}) - \beta\log q_\phi(\mathbf{z}|\mathbf{x}) + \beta\log p(\mathbf{z})] + \lambda(\mathbf{x})(q_\phi(\mathbf{z}|\mathbf{x}) - 1)\right] \tag{C.25}$$

For brevity, going forward $p_\theta(\mathbf{x}|\mathbf{z}) = p$, $p(\mathbf{z}) = \pi$, $q_\phi(\mathbf{z}|\mathbf{x}) = q$. We also view $\mathcal{L}$ as depending on q, p directly.

To proceed with calculus of variations, we substitute $q \to q + \epsilon$, where ϵ is a small function that goes to zero appropriately fast for large $\mathbf{x}, \mathbf{z}$. Thus we expand $\mathcal{L}$ to first order in q to find $\frac{\delta\mathcal{L}}{\delta q}$. The form of q for which this gradient is zero gives us the optimum q for this functional.

$$\mathcal{L}_\beta(q+\epsilon) = \int \mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{z}\,\rho(\mathbf{x})\big[(q+\epsilon)[\log p - \beta\log(q+\epsilon) + \beta\log\pi] + \lambda(\mathbf{x})(q+\epsilon-1)\big] \quad \text{(C.26)}$$

Recall that $\log(1+x) \approx x$ to first order. Thus $\log(q+\epsilon) \approx \log q + \frac{\epsilon}{q}$ to first order. So,

$$\mathcal{L}_\beta(q+\epsilon) = \int \mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{z}\,\rho(\mathbf{x})q\big[\log p - \beta\log q + \beta\log\pi - \beta\frac{\epsilon}{q}\big] \quad \text{(C.27)}$$

$$+ \int \mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{z}\,\rho(\mathbf{x})\epsilon\big[\log p - \beta\log q + \beta\log\pi - \beta\frac{\epsilon}{q}\big] \quad \text{(C.28)}$$

$$+ \int \mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{z}\,\rho(\mathbf{x})\lambda(\mathbf{x})(q-1) + \int \mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{z}\,\rho(\mathbf{x})\lambda(\mathbf{x})\epsilon + \int \mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{z}\,\rho(\mathbf{x})O(\epsilon^2). \quad \text{(C.29)}$$

Rearranging we find

$$\mathcal{L}_\beta(q+\epsilon) = \mathcal{L}_\beta(q) + \int \mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{z}\,\rho(\mathbf{x})\epsilon\big[\log p - \beta\log q + \beta\log\pi - \beta + \lambda(\mathbf{x})\big] + \int \mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{z}\,\rho(\mathbf{x})O(\epsilon^2) \quad \text{(C.30)}$$

$$= \mathcal{L}_\beta(q) + \int \mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{z}\,\frac{\delta\mathcal{L}_\beta}{\delta q}\epsilon + \int \mathrm{d}\mathbf{x}\,\mathrm{d}\mathbf{z}\,\rho(\mathbf{x})O(\epsilon^2) \quad \text{(C.31)}$$

At the optimum value of q the functional will have vanishing functional derivative $\frac{\delta\mathcal{L}_\beta}{\delta q}$, so

$$\log p - \beta\log q + \beta\log\pi - \beta + \lambda(\mathbf{x}) = 0, \quad \text{(C.32)}$$

$$\log q = \frac{1}{\beta}\log p + \log\pi + C(\mathbf{x}). \quad \text{(C.33)}$$

Exponentiating we find the optimal q to be

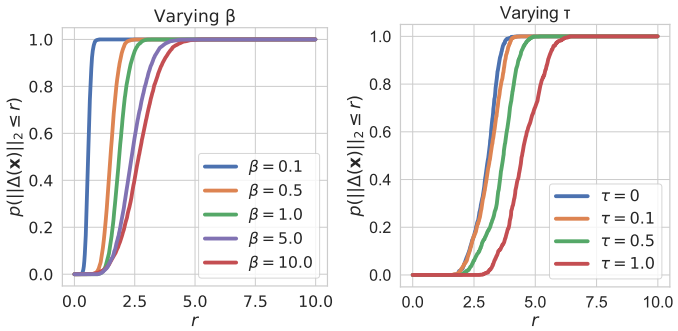
$$q_\phi(\mathbf{z}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})}p_\theta(\mathbf{x}|\mathbf{z})^{\frac{1}{\beta}}p(\mathbf{z}), \quad \text{(C.34)}$$

where Z is an appropriate normalising constant. This completes the proof. ■

C.4 Empirical Calculation of the Bounds

C.4.1 Estimating the minimum r

C.4.2 Results



(a) (b)

Figure C.4.2: Here we show that the minimum r for which $p(||\Delta(\mathbf{x})||_2 \leq r) = 0.5$ increases with β and τ, where β is the penalty applied to the KL in β-VAEs and τ is an offset added to the encoder standard deviation $\boldsymbol{\sigma}_\phi(\mathbf{x})$. This probability, estimated as detailed below in Appendix C.4.3, increases with r, but increases more slowly for large β (a) and large τ (b). In such models the encoding process has higher variance resulting in a greater spread of reconstructions, confirming Proposition C.1.1 in Appendix A that the minimum r for r-robustness increases with the encoder variance.

C.4.3 Algorithm

Algorithm 1 Estimating r

Result: r such that $p(||\Delta(\mathbf{x})||_2 \leq r) > 0.5$

m, *step*, *samples*, $\mathbf{x}$, $r \leftarrow 0$, $p(||\Delta(\mathbf{x})||_2 \leq r) \leftarrow 0$ **while** $p(||\Delta(\mathbf{x})||_2 \leq r) < m$ **do**
  $d \leftarrow \{\}$ **for** $i \leftarrow 1$ **to** *samples* **by** 1 **do**
    $\mathbf{s} \sim \mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{x}), \boldsymbol{\sigma}_\phi(\mathbf{x}))$ $s_d \leftarrow ||g_\theta(\mathbf{s}) - g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))||_2$ d.insert(s_d)
  end
  $r \leftarrow r + step$ $p(||\Delta(\mathbf{x})||_2 \leq r) \leftarrow \frac{\text{Sum}(d<r)}{nsamples}$
end

C.5 Estimating the Margin of Robustness

Algorithm 2 Estimating $R^r_{\mathcal{X}}(\mathbf{x})$

Result: $R^r_{\mathcal{X}}(\mathbf{x})$ such that $p(||\Delta(\mathbf{x}, \boldsymbol{\delta}_x)||_2 \leq r) > 0.5$

step, *samples*, $\mathbf{x}$, r, $p(||\Delta(\mathbf{x}, \boldsymbol{\delta}_x)||_2 \leq r) \leftarrow 0$, $R^r_{\mathcal{X}}(\mathbf{x}) \leftarrow 10$, *restarts* $\leftarrow 5$ **while** $p(||\Delta(\mathbf{x}, \boldsymbol{\delta}_x)||_2 \leq r) < 0.5$ **do**

  for $j \leftarrow 1$ **to** *restarts* **by** 1 **do**

    $d \leftarrow \{\}$ **for** $i \leftarrow 1$ **to** *samples* **by** 1 **do**

      $\boldsymbol{\delta}_x \leftarrow$ max damage attack constrained to the norm $R^r_{\mathcal{X}}(\mathbf{x})$ $\mathbf{s} \sim \mathcal{N}(\boldsymbol{\mu}_\phi(\mathbf{x} + \boldsymbol{\delta}_x), \boldsymbol{\sigma}_\phi(\mathbf{x} + \boldsymbol{\delta}_x))$

      $s_d \leftarrow ||g_\theta(\mathbf{s}) - g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))||_2$ d.insert(s_d)

    end

    $R^r_{\mathcal{X}}(\mathbf{x}) \leftarrow R^r_{\mathcal{X}}(\mathbf{x}) - step$ $p(||\Delta(\mathbf{x}, \boldsymbol{\delta}_x)||_2 \leq r) \leftarrow \frac{\text{Sum}(d<r)}{nsamples}$

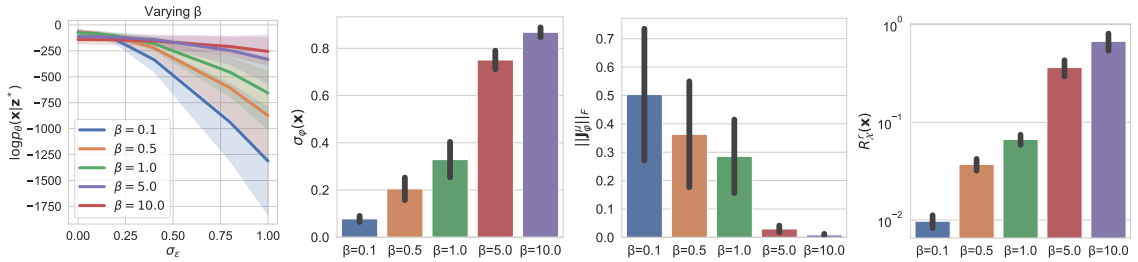
  end

end

C.6 -VAE Sensitivity Experiments



(a) $\beta = 0.1$ (b) $\beta = 0.5$ (c) $\beta = 1$ (d) $\beta = 5$ (e) $\beta = 10$

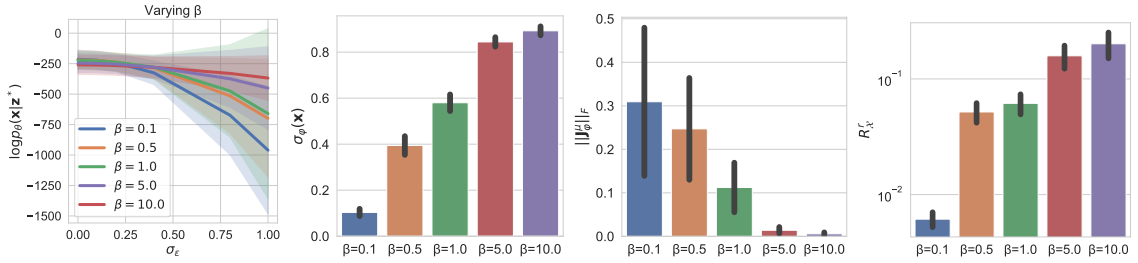


(f) VAE Sensitivity (g) Encoder Variance (h) Encoder Jacobian (i) $R^r_{\mathcal{X}}(\mathbf{x})$ bound

Figure C.6.3: Here we illustrate that β-VAEs, trained MNIST, with higher β penalties generalise better and are less sensitive to input perturbations. The first row (a)-(e) shows samples drawn from the latent space prior that are then fed through the VAE decoder. It is clear that as β increases, so too does the quality of generated samples. (f) shows the sensitivity of the VAE to input perturbations. We add zero-mean Gaussian noise of variance σ_ϵ^2 to the VAE input to form a noisy input $\mathbf{x}^*$ and embedding $\mathbf{z}*$. We then measure the likelihood of the original point $\mathbf{x}$ under this noisy embedding. σ_ϵ^2 is thus an approximation of the margin of robustness of the VAE, if the VAE's likelihood does not change even for high variance noise, it must have a large margin of robustness ($R^r_{\mathcal{X}}(\mathbf{x})$). The likelihood of $\mathbf{x}$ is quasi constant, under increasing noise variance, for high values of β. This supports our analysis that such models have higher $R^r_{\mathcal{X}}(\mathbf{x})$. Figures (g) and (h) show that the encoder variance and that the norm of the encoder Jacobian ($||\mathbf{J}^{\mu}_{\phi}(\mathbf{x})||_F$) increase as β increases, supporting our analysis that the changes in these values underpin the robustness observed. In (i) we calculate the bound for $R^r_{\mathcal{X}}(\mathbf{x})$ from Theorem 5.1 where we ignore higher order terms. We select r such that $p_{A^r}(\mathbf{x}) = 0.9$, which is a relatively strict metric for robustness. In (f-i) confidence intervals correspond to the standard deviations of values over the entire MNIST dataset. Taken as a whole these experiments support our analysis that the margin $R^r_{\mathcal{X}}(\mathbf{x})$ increases with β as in Theorem 5.2, in conjuction with the norm of the encoder Jacobian and the encoder variance, supporting Theorem 5.1.

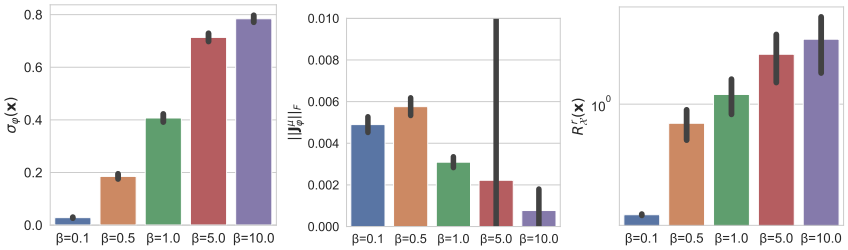


(a) $\beta = 0.1$ (b) $\beta = 0.5$ (c) $\beta = 1$ (d) $\beta = 5$ (e) $\beta = 10$



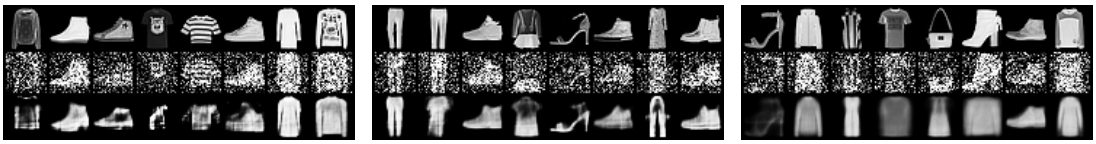
(f) VAE Sensitivity (g) Encoder Variance (h) Encoder Jacobian (i) $R^r_{\mathcal{X}}(\mathbf{x})$ bound

Figure C.6.4: Here we illustrate that β-VAEs, trained on fashion-MNIST, with higher β penalties generalise better and are less sensitive to input perturbations. The first row (a)-(e) shows samples drawn from the latent space prior that are then fed through the VAE decoder. It is clear that as β increases, so too does the quality of generated samples. (f) shows the sensitivity of the VAE to input perturbations. We add zero-mean Gaussian noise of variance σ^2_ϵ to the VAE input to form a noisy input $\mathbf{x}^*$ and embedding $\mathbf{z}*$. We then measure the likelihood of the original point $\mathbf{x}$ under this noisy embedding. σ^2_ϵ is thus an approximation of the margin of robustness of the VAE, if the VAE's likelihood does not change even for high variance noise, it must have a large margin of robustness ($R^r_{\mathcal{X}}(\mathbf{x})$). The likelihood of $\mathbf{x}$ is quasi constant, under increasing noise variance, for high values of β. This supports our analysis that such models have higher $R^r_{\mathcal{X}}(\mathbf{x})$. Figures (g) and (h) show that the encoder variance and that the encoder Jacobian norm ($||\mathbf{J}^{\mu}_{\phi}(\mathbf{x})||_F$) increase as β increases, supporting our analysis that the changes in these values underpin the robustness observed. In (i) we calculate the bound for $R^r_{\mathcal{X}}(\mathbf{x})$ from Theorem 5.1 where we ignore higher order terms. We select r such that $p_{A^r}(\mathbf{x}) = 0.9$, which is a relatively strict metric for robustness. In (f-i) confidence intervals correspond to the standard deviations of values over the entire fashion-MNIST dataset. Taken as a whole these experiments support our analysis that the margin $R^r_{\mathcal{X}}(\mathbf{x})$ increases with β as in Theorem 5.2, in conjuction with the norm of the encoder Jacobian and the encoder variance, supporting Theorem 5.1.



(a) Encoder Variance (b) Encoder Jacobian (c) $R^r_{\mathcal{X}}(\mathbf{x})$ bound

Figure C.6.5: Here we illustrate that β-VAEs, trained on CIFAR10, with higher β have larger margins of robustness. Figures (a) and (b) show that the encoder variance and that the encoder Jacobian norm ($||\mathbf{J}^\mu_\phi(\mathbf{x})||_F$) increase as β increases, supporting our analysis that the changes in these values underpin the robustness observed. In (i) we calculate the bound for $R^r_{\mathcal{X}}(\mathbf{x})$ from Theorem 5.1 where we ignore higher order terms. We select r such that $p_{A^r}(\mathbf{x}) = 0.9$, which is a relatively strict metric for robustness. In (a-c) confidence intervals correspond to the standard deviations of values over the entire dataset. Taken as a whole these experiments support our analysis that the margin $R^r_{\mathcal{X}}(\mathbf{x})$ increases with β as in Theorem 5.2, in conjuction with the norm of the encoder Jacobian and the encoder variance, supporting Theorem 5.1.



(a) $\beta = 0.1$ (b) $\beta = 1$ (c) $\beta = 10$

Figure C.6.6: We show reconstructions of noisy data for VAEs trained with $\beta \in \{0.1, 1, 10\}$ on Fashion-MNIST. The first row corresponds to the original image, the second to noised a image $\mathbf{x} + \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, (0.5^2)\mathbf{I})$. Clearly larger β models are less sensitive to noise, supporting our analysis that increasing β increases the margin of robustness to perturbations.

C.7 Network Hyperparameters

All networks used the same hyperparameters. Namely networks were trained for 100 epochs with the Adam optimizer, with a learning rate of 0.001 and a batch size of 512.

For MNIST and fashion-MNIST networks for the encoder variance and encoder mean were two hidden layer multi-layer perceptrons (MLPs) with 400 units per layer, which shared their first layer. Similarly the decoder was a two layer MLP with 400 units per layer. For these datasets we used a latent space size of 20.

For CIFAR10 we used 4-layer MLPs with 400 units per layer for the encoder and decoder networks and used a 64-dimensional latent space.

APPENDIX D

APPENDIX: VARIATIONAL AUTOENCODERS: A HARMONIC PERSPECTIVE

D.1 Technical Proofs . . . 175

D.1.1 Proof of Proposition 6.1 . . . 176

D.1.2 Proof of Proposition 6.2 . . . 176

D.1.3 Proof of Theorem 6.3 . . . 177

D.1.4 Proof of Proposition 6.4 . . . 179

D.2 Multivariate Hermite Coefficients for a Gaussian measure with full Covariance . . . 180

D.3 Implementation Details . . . 181

D.4 Additional Results . . . 181

D.4.1 Fourier Transform of Gaussian Measure . . . 182

D.4.2 Fourier Spectrum of Multivariate VAEs . . . 182

D.4.3 Gaussian Spaces and VAE Lipschitzness . . . 183

D.1 Technical Proofs

Before we begin the proofs we give an alternative definition of the Hermite polynomials which uses the k^{th} divergence operator (δ^k, (Janson, 1997)). This definition greatly simplifies some of our later proofs.

$$H_k = \delta^k 1(x), \tag{D.1}$$

where 1 is shorthand for the function that is identically 1. The divergence operator is defined using $\text{Dom}(\delta^k)$, which is the subset of functions $g \in L_2(\mathbb{R}, \gamma)$ for which there exists a $c > 0$ such that for all functions for which the derivative up to degree k obeys $f^{(k)} \in L_2(\mathbb{R}, \gamma)$:

$$\left| \int_{\mathbb{R}} f^{(k)}(x) g(x) d\gamma(x) \right| \leq c \sqrt{\left| \int_{\mathbb{R}} f^{(k)}(x) d\gamma(x) \right|}.$$

$1(x)$ for example, as defined in Equation (D.1) is in $\text{Dom}(\delta^k)$. The k^{th} divergence can then be defined as follows. If $g \in \text{Dom}(\delta^k)$ then $\delta^k g$ is the unique element of $L_2(\mathbb{R}, \gamma)$ such that for all functions for which the derivative up to degree $\boldsymbol{\alpha}$ obeys $f^{(k)} \in L_2(\mathbb{R}, \gamma)$ we have:

$$\int_{\mathbb{R}} f(x) \delta^k g(x) d\gamma(x) = \int_{\mathbb{R}} f^{(k)}(x) d\gamma(x). \tag{D.2}$$

For the general Gaussian space, this becomes by substitution of variables:

$$H_k = \delta^k 1(\hat{x}), \qquad \hat{x} = (x - \mu)/\sigma. \tag{D.3}$$

In the multivariate case, for the general Gaussian space with diagonal standard deviation, we have that $\mathcal{H}_{\boldsymbol{\alpha}} = \delta^{\boldsymbol{\alpha}} 1(\hat{\mathbf{x}})$, $\hat{\mathbf{x}} = (\mathbf{x} - \boldsymbol{\mu})\mathbf{S}^{-1}$, where δ now is the vector-valued divergence operator indexed by the multi-index $\boldsymbol{\alpha}$.

D.1.1 Proof of Proposition 6.1

Proof. Due to the scaling and centering relation for Gaussians a simple change of variables $\hat{x} = (x - \mu)/\sigma$ in Equation (6.2) gives us the first part of the proof. Thus

$$\mathbb{E}_{\gamma(\hat{x})}[H_k(\hat{x})H_m(\hat{x})] = k!\mathbf{1}\{m = k\},$$

and thus $\left\{\frac{1}{k!}H_k : k \geq 0\right\}$ is orthonormal in $L_2(\mathbb{R}, \mathcal{N}(\mu, \sigma^2))$ because $\gamma(\hat{x}) = \mathcal{N}(\mu, \sigma^2)$.

We know that for $k \geq 0$ the polynomial H_k has degree k, hence it suffices to demonstrate that the mononomials of degree k, $\{y^k : k \in \mathbb{N}, \hat{x} = (x - \mu)/\sigma\}$ are dense in $L_2(\mathbb{R}, \gamma(\hat{x}))$ to show that $\left\{\frac{1}{\boldsymbol{\alpha}!}H_k : k \geq 0\right\}$ is a basis for $L_2(\mathbb{R}, \gamma(\hat{x}))$.

The Elementary Hahn-Banach theorem shows that if $g \in L_2(\mathbb{R}, \gamma)$ is such that $\int_{\mathbb{R}} g(\hat{x})\hat{x}^k d\gamma(\hat{x}) = 0, \forall k \in \mathbb{N}^+$ then it suffices to show that $g = 0$ almost everywhere to demonstrate that the mononomials $\hat{x}^k$ are a dense subspace of $L_2(\mathbb{R}, \gamma(\hat{x}))$. This proof generally follows that used for the standard Gaussian measure. Assume we have a g that satisfies $\int_{\mathbb{R}} g(\hat{x})\hat{x}^k d\gamma(\hat{x}) = 0$. By the series expansion of the exponential function we know that for all $t \in \mathbb{R}$

$$\int_{\mathbb{R}} g(\hat{x})e^{ity} d\gamma(\hat{x}) = \lim_{m\to\infty} \sum_{k=0}^{m} \frac{(it)^k}{k!} \int_{\mathbb{R}} g(\hat{x})\hat{x}^k d\gamma(\hat{x}) = 0.$$

Thus we have that $\int_{\mathbb{R}} g(\hat{x})e^{ity} d\gamma(\hat{x}) = 0, \forall t$. Because e^{ity} is injective we have that $g(\hat{x}) = 0, \forall \hat{x} \in \mathbb{R}$.

■

D.1.2 Proof of Proposition 6.2

Proof. We begin with the univariate case, from which we build up to the multivariate case. Here we use the divergence operator definition of the Hermite polynomials, seen in Equation (D.1). In our case by substitution of variables $H_k(\hat{x}) = \delta^k 1(\hat{x})$ and we can express x as $x = y\sigma + \mu$, thus:

$$\int_{\mathbb{R}} f(y\sigma + \mu)H_k(\hat{x}) d\gamma(\hat{x}) = \int_{\mathbb{R}} f(y\sigma + \mu)\delta^k 1(\hat{x}) d\gamma(\hat{x}) = \int_{\mathbb{R}} \sigma^k f^k(\hat{x}\sigma + \mu) d\gamma(\hat{x}).$$

As mentioned in the main paper, in the multivariate case, the divergence operator is an almost exact analogue to that of the univariate case. Due to the tensorisation of bases to form $\mathcal{H}$ the indices α simply become a multi-index and we recover that $\hat{f}(\boldsymbol{\alpha}) = \mathbb{E}_{\gamma(\hat{\mathbf{x}})}\left[\mathbf{S}^{\boldsymbol{\alpha}} f^{(\boldsymbol{\alpha})}(\mathbf{x})\right]$.

■

D.1.3 Proof of Theorem 6.3

Proof. By Proposition 6.2 we know that we can express a function $f \in L_2(\mathbb{R}^n, \mathcal{N}(\boldsymbol{\mu}, \mathbf{S}^2))$ that is infinitely differentiable as:

$$f = \sum_{\boldsymbol{\alpha}\in\mathbb{N}^n} \frac{1}{\boldsymbol{\alpha}!}(\text{diag}(\mathbf{S}))^{\boldsymbol{\alpha}}\, \mathbb{E}_{\gamma(\hat{\mathbf{x}})}\left[f^{(\boldsymbol{\alpha})}(\mathbf{x})\right] \mathcal{H}_{\boldsymbol{\alpha}}, \qquad \gamma(\hat{\mathbf{x}}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{S}^2).$$

Now note that the expectation of f is simply the first Hermite coefficient with degree 0:

$$\mathbb{E}_{\gamma(\hat{\mathbf{x}})}[f] = \sum_{|\boldsymbol{\alpha}|=0} \frac{1}{\boldsymbol{\alpha}!}(\text{diag}(\mathbf{S}))^{\boldsymbol{\alpha}}\, \mathbb{E}_{\gamma(\hat{\mathbf{x}})}\left[f^{(\boldsymbol{\alpha})}(\mathbf{x})\right] \mathcal{H}_{\boldsymbol{\alpha}}, \qquad \gamma(\hat{\mathbf{x}}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{S}^2).$$

Now taking the variance of f:

$$\begin{aligned}
\text{Var(f)} &= \mathbb{E}_{\gamma(\hat{\mathbf{x}})}\left[\left(f - \mathbb{E}_{\gamma(\hat{\mathbf{x}})}[f]\right)^2\right] \\
&= \mathbb{E}_{\gamma(\hat{\mathbf{x}})}\left[\left(\sum_{\boldsymbol{\alpha}\in\mathbb{N}^n} \frac{1}{\boldsymbol{\alpha}!}(\text{diag}(\mathbf{S}))^{\boldsymbol{\alpha}}\, \mathbb{E}_{\gamma(\hat{\mathbf{x}})}\left[f^{(\boldsymbol{\alpha})}(\mathbf{x})\right] \mathcal{H}_{\boldsymbol{\alpha}} - \sum_{|\boldsymbol{\alpha}|=0} \frac{1}{\boldsymbol{\alpha}!}(\text{diag}(\mathbf{S}))^{\boldsymbol{\alpha}}\, \mathbb{E}_{\gamma(\hat{\mathbf{x}})}\left[f^{(\boldsymbol{\alpha})}(\mathbf{x})\right] \mathcal{H}_{\boldsymbol{\alpha}}\right)^2\right] \\
&= \mathbb{E}_{\gamma(\hat{\mathbf{x}})}\left[\left(\sum_{|\boldsymbol{\alpha}|\geq 1} \frac{1}{\boldsymbol{\alpha}!}(\text{diag}(\mathbf{S}))^{\boldsymbol{\alpha}}\, \mathbb{E}_{\gamma(\hat{\mathbf{x}})}\left[f^{(\boldsymbol{\alpha})}(\mathbf{x})\right] \mathcal{H}_{\boldsymbol{\alpha}}\right)^2\right].
\end{aligned}$$

Recall that the bases $\mathcal{H}_{\boldsymbol{\alpha}}$ are orthogonal such that

$$\mathbb{E}_{\gamma(\hat{\mathbf{x}})}[\mathcal{H}_{\boldsymbol{\alpha}}(\hat{\mathbf{x}})\mathcal{H}_m(\hat{\mathbf{x}})] = \boldsymbol{\alpha}!\{m = \boldsymbol{\alpha}\}.$$

As such we have that:

$$\begin{aligned}\text{Var(f)} &= \mathbb{E}_{\gamma(\hat{\mathbf{x}})}\left[\left(\sum_{|\boldsymbol{\alpha}|\geq 1}\frac{1}{\boldsymbol{\alpha}!}(\text{diag}(\mathbf{S}))^{\boldsymbol{\alpha}}\,\mathbb{E}_{\gamma(\hat{\mathbf{x}})}\left[f^{(\boldsymbol{\alpha})}(\mathbf{x})\right]\mathcal{H}_{\boldsymbol{\alpha}}\right)^2\right] \\ &= \mathbb{E}_{\gamma(\hat{\mathbf{x}})}\left[\sum_{|\boldsymbol{\alpha}|\geq 1}\left(\frac{1}{\boldsymbol{\alpha}!}(\text{diag}(\mathbf{S}))^{\boldsymbol{\alpha}}\,\mathbb{E}_{\gamma(\hat{\mathbf{x}})}\left[f^{(\boldsymbol{\alpha})}(\mathbf{x})\right]\mathcal{H}_{\boldsymbol{\alpha}}\right)^2\right] \\ &= \sum_{|\boldsymbol{\alpha}|\geq 1}\frac{(\text{diag}(\mathbf{S}))^{2\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}\left|\mathbb{E}_{\gamma(\hat{\mathbf{x}})}\left[f^{(\boldsymbol{\alpha})}(\mathbf{x})\right]\right|^2.\end{aligned}$$

This completes the first part of the proof.

For the second part we focus on terms of the form

$$\mathbb{E}_{\gamma(\hat{\mathbf{x}})}\left[f^{(\boldsymbol{\alpha})}(\mathbf{x})\right] = \int_{\mathbb{R}^n} f^{(\boldsymbol{\alpha})}(\hat{\mathbf{x}}\mathbf{S}+\boldsymbol{\mu})d\gamma(\hat{\mathbf{x}}) = \int_{\mathbb{R}^n} f^{(\boldsymbol{\alpha})}(\hat{\mathbf{x}}\mathbf{S}+\boldsymbol{\mu})\gamma(\hat{\mathbf{x}})d\hat{\mathbf{x}}.$$

Both f and γ are L_2 integrable with respect to the Lebesgue measure. As such we can directly apply Plancherel's theorem to demonstrate that

$$\begin{aligned}\mathbb{E}_{\gamma(\hat{\mathbf{x}})}\left[f^{(\boldsymbol{\alpha})}(\mathbf{x})\right] &= \int_{\mathbb{R}^n} f^{(\boldsymbol{\alpha})}(\hat{\mathbf{x}}\mathbf{S}+\boldsymbol{\mu})\gamma(\hat{\mathbf{x}})d\hat{\mathbf{x}} \\ &= \int_{\mathbb{R}^n} f^{(\boldsymbol{\alpha})}(\mathbf{x})\gamma((\mathbf{x}-\boldsymbol{\mu})\mathbf{S}^{-1})d\mathbf{x} \\ &= \int_{\mathbb{R}^n} (i\boldsymbol{\omega})^{\boldsymbol{\alpha}}\mathcal{F}(\boldsymbol{\omega})\overline{\det(\mathbf{S})\mathcal{G}(\boldsymbol{\omega}\mathbf{S})e^{-i\boldsymbol{\omega}\boldsymbol{\mu}\mathbf{S}^{-1}}}d\boldsymbol{\omega} \\ &= \int_{\mathbb{R}^n} (i\boldsymbol{\omega})^{\boldsymbol{\alpha}}\mathcal{F}(\boldsymbol{\omega})\overline{\mathcal{P}(\boldsymbol{\omega})}d\boldsymbol{\omega}.\end{aligned}$$

where $\mathcal{P}$ is the Fourier transform of the Gaussian measure $\mathcal{N}(\mu,\sigma^2)$ given by $\overline{\mathcal{P}(\boldsymbol{\omega})} = \det(\mathbf{S})\mathcal{G}(\boldsymbol{\omega})e^{-i\boldsymbol{\omega}\boldsymbol{\mu}\mathbf{S}^{-1}}$ (where $\mathcal{G}$ is the Fourier transform of the standard Gaussian measure γ) and $\mathcal{F}$ is the Fourier transform of f. This stems from the fact that the Fourier transform of the $\boldsymbol{\alpha}^{\text{th}}$ derivative of a function is simply $(i\boldsymbol{\omega})^{\boldsymbol{\alpha}}\mathcal{F}(\boldsymbol{\omega})$

From this we obtain that

$$\begin{aligned}\text{Var(f)} &= \sum_{|\boldsymbol{\alpha}|\geq 1}\frac{(\text{diag}(\mathbf{S}))^{2\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}\left|\int_{\mathbb{R}^n}(i\boldsymbol{\omega})^{\boldsymbol{\alpha}}\mathcal{F}(\boldsymbol{\omega})\overline{\mathcal{P}(\boldsymbol{\omega})}d\boldsymbol{\omega}\right|^2 \\ &= \sum_{|\boldsymbol{\alpha}|\geq 1}\frac{(\text{diag}(\mathbf{S}))^{2\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}\left|\int_{\mathbb{R}^n}(\boldsymbol{\omega})^{\boldsymbol{\alpha}}\mathcal{F}(\boldsymbol{\omega})\overline{\mathcal{P}(\boldsymbol{\omega})}d\boldsymbol{\omega}\right|^2, \boldsymbol{\alpha}\in\mathbb{N}^n.\end{aligned}$$

■

D.1.4 Proof of Proposition 6.4

Proof. We begin with a proof for a univariate Gaussian space. We once again use the definition of the Hermite polynomials in terms of the divergence operator. Note that $H_\alpha = \delta H_{\alpha-1}$. The variance of a function $f \in L_2(\mathbb{R}, \mathcal{N}(\mu, \sigma^2))$ can be expressed as the sum of Hermite coefficients squared:

$$\mathrm{Var}(f) = \sum_{\alpha=1}^{\infty} \frac{1}{\alpha!} |\hat{f}(\alpha)|^2 = \sum_{\alpha=1}^{\infty} \frac{1}{\alpha!} \left| \mathbb{E}_{\gamma(\hat{x})} \left[f(\sigma\hat{x} + \mu) H_\alpha(\hat{x}) \right] \right|^2 . \tag{D.4}$$

Using the definition of the divergence operator we have that:

$$\mathrm{Var}(f) = \sum_{\alpha=1}^{\infty} \frac{1}{\alpha!} \left| \mathbb{E}_{\gamma(\hat{x})} \left[\sigma f^{(1)}(\sigma\hat{x} + \mu) H_{\alpha-1}(\hat{x}) \right] \right|^2 \tag{D.5}$$

$$\leq \sum_{\alpha=1}^{\infty} \frac{1}{(\alpha-1)!} \left| \mathbb{E}_{\gamma(\hat{x})} \left[\sigma f^{(1)}(\sigma\hat{x} + \mu) H_{\alpha-1}(\hat{x}) \right] \right|^2 \tag{D.6}$$

$$= \sum_{\alpha=0}^{\infty} \frac{\sigma^2}{\alpha!} \left| \mathbb{E}_{\gamma(\hat{x})} \left[f^{(1)}(\sigma\hat{x} + \mu) H_\alpha(\hat{x}) \right] \right|^2 . \tag{D.7}$$

As $f^{(1)}$ is a member of the Gaussian space, the last equation is simply the dot product of $f^{(1)}$ with itself, weighted by σ^2. Thus:

$$\mathrm{Var}(f) = \sigma^2 \langle f^{(1)}, f^{(1)} \rangle. \tag{D.8}$$

By definition of the Lipschitz constant:

$$L \leq \sup_{x, x' \in \mathbb{R}^\kappa, x \neq x'} \frac{|f(x) - f(x')|}{\|x - x'\|}.$$

We then have

$$\mathrm{Var}(f) \leq \sigma^2 L^2. \tag{D.9}$$

For the multivariate case, we work in reverse. We know that $L^2 \geq \|Df\|^2$ where Df

is the Jacbobian of the function with respect to the input such that

$$\left[\|Df\|^2\right] = \sum_{|\boldsymbol{\alpha}|=1} < f^{(\boldsymbol{\alpha})}, f^{(\boldsymbol{\alpha})} >$$

As by assumption each $f^{(\boldsymbol{\alpha})}, |\boldsymbol{\alpha}| = 1$ is a member of the Gaussian space we have that

$$\|Df\|^2 = \sum_{|\boldsymbol{\alpha}|=1} \sum_{\beta \in \mathbb{N}^n} \frac{1}{\beta!} \left| \mathbb{E}_{\gamma(\hat{\mathbf{x}})} \left[f^{(\boldsymbol{\alpha})}(\mathbf{S}\hat{\mathbf{x}} + \boldsymbol{\mu}) \mathcal{H}_\beta(\hat{\mathbf{x}}) \right] \right|^2 \tag{D.10}$$

$$= \sum_{|\boldsymbol{\alpha}|=1} \sum_{\beta \in \mathbb{N}^n} \frac{1}{\beta!} \left| \mathbb{E}_{\gamma(\hat{\mathbf{x}})} \left[\mathbf{S}^{-1} f(\mathbf{S}\hat{\mathbf{x}} + \boldsymbol{\mu}) \delta^{\boldsymbol{\alpha}} \mathcal{H}_\beta(\hat{\mathbf{x}}) \right] \right|^2 \tag{D.11}$$

$$= \sum_{|\boldsymbol{\alpha}|=1} \sum_{\beta \in \mathbb{N}^n} \frac{1}{\beta!} \left| \mathbb{E}_{\gamma(\hat{\mathbf{x}})} \left[\mathbf{S}^{-1} f(\mathbf{S}\hat{\mathbf{x}} + \boldsymbol{\mu}) \mathcal{H}_{\beta + \boldsymbol{\alpha}}(\hat{\mathbf{x}}) \right] \right|^2 \tag{D.12}$$

$$\geq \sum_{|\beta| \geq 1} \frac{1}{\beta!} \left| \mathbb{E}_{\gamma(\hat{\mathbf{x}})} \left[\mathbf{S}^{-1} f(\mathbf{S}\hat{\mathbf{x}} + \boldsymbol{\mu}) \mathcal{H}_\beta(\hat{\mathbf{x}}) \right] \right|^2, \beta \in \mathbb{N}^n. \tag{D.13}$$

This last term is simply $\mathrm{Var}(f)/\|\mathbf{S}\|^2$ (see the proof of Theorem 6.3 for why this is). Thus we have that $L^2\|\mathbf{S}\|^2 \geq \mathrm{Var}(f)$. This concludes the proof.

■

D.2 Multivariate Hermite Coefficients for a Gaussian measure with full Covariance

Let our Gaussian space be equipped with Gaussian measure $\mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$, where $\mathbf{C}$ is a full covariance matrix. By definition $\mathbf{C}$ is symmetric and positive definite, thus there exists an orthogonal matrix $\mathbf{O}$ such that:

$$\mathbf{O}\mathbf{C}\mathbf{O}^T = \mathbf{D},$$

where $\mathbf{D}$ is some diagonal matrix. Let $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$, we know that:

$$\hat{\mathbf{x}} = (\mathbf{x} - \boldsymbol{\mu})\mathbf{O}^T\mathbf{D}^{-\frac{1}{2}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

By substitution of variables for the Hermite polynomials in the case of a diagonal covariance, we have that the Hermite polynomials for the measure $\mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$ are given by:

$$\mathcal{H}_{\boldsymbol{\alpha}}((\mathbf{x}-\boldsymbol{\mu})\mathbf{O}^T\mathbf{D}^{-\frac{1}{2}}) = \mathcal{H}_{\boldsymbol{\alpha}}(\hat{\mathbf{x}}) = \prod_i H_{\alpha_i}(\hat{x}_i).$$

D.3 Implementation Details

The architecture of VAEs with dense layers is presented in text. For convolutional networks the encoder layers had the following number of filters in order: $\{64, 64, 128, 128, 512\}$.

The mean and variance of the amortised posteriors are the output of dense layers acting on the output of the purely convolutional network, where the number of neurons in these layers is equal to the dimensionality of the latent space $\mathcal{Z}$.

Similarly, for the decoders ($p(\mathbf{x}|\mathbf{z})$) of all our models we also used purely convolutional networks with 6 deconvolutional layers. The layers had the following number of filters in order: $\{512, 128, 128, 64, 64, 3\}$. The mean of the likelihood $p(\mathbf{x}|\cdot)$ was directly encoded by the final de-convolutional layer. The variance of the decoder was fixed to 0.1.

To train models we used ADAM (Kingma and Lei Ba, 2015) with default parameters, a learning rate of 0.001, and a batch size of 1024. All data was preprocessed to fall on the interval -1 to 1.

All models were trained with an *Azure Cloud Standard NC6* machine with a single *NVIDIA Tesla K80*.

D.4 Additional Results

D.4.1 Fourier Transform of Gaussian Measure

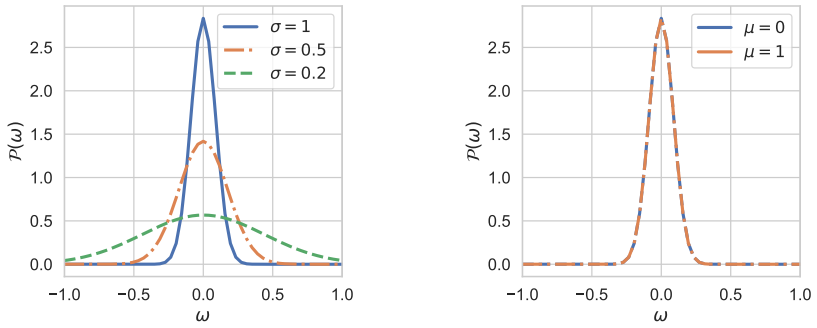


Figure D.4.1: [left] Fourier transform of the Gaussian measure $\mathcal{N}(0, \sigma^2)$ for varying σ. [right] Fourier transform of the Gaussian measure $\mathcal{N}(\mu, 1)$ for varying μ.

D.4.2 Fourier Spectrum of Multivariate VAEs

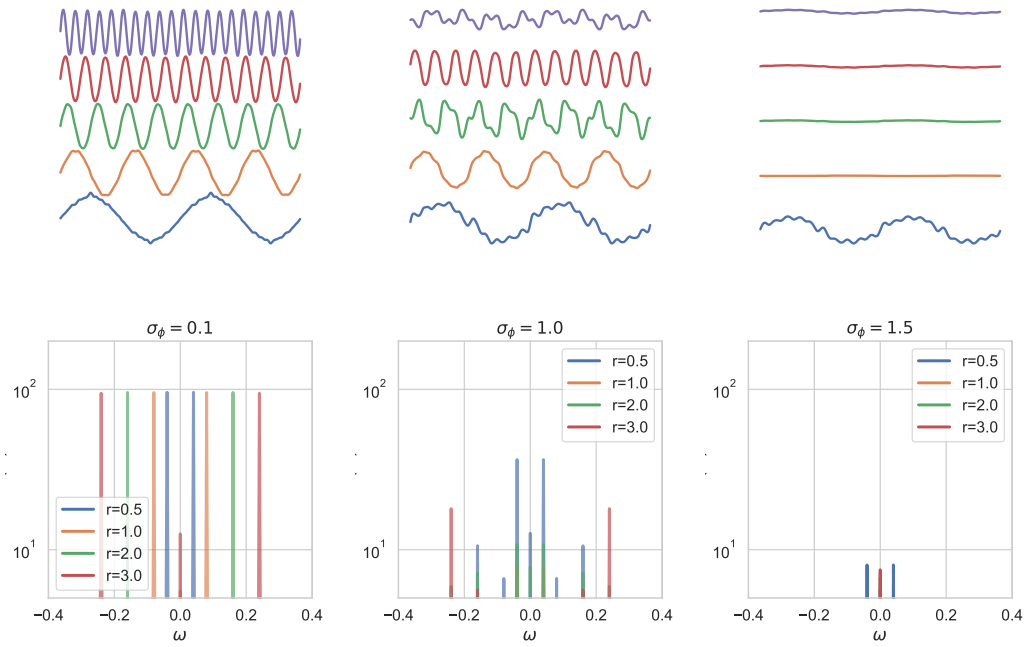


Figure D.4.2: We train VAEs with 3-dense layers (each with 256 units), with ReLU activation functions in their networks, on multivariate data consisting of 5 sinusoids $y = \sin(2\pi r t)$, $r \in \{0.5, 1.0, 2.0, 3.0, 5.0\}$, $t \in [-1, 1]$, with a 1 dimension latent space $\mathcal{Z}$. As before we fix the encoder σ_ϕ. **[top row]** We show plots of the regressed sinusoids for different values of σ_ϕ. **[bottom row]** Below we show an FFT for each regressed sinusoid. As σ_ϕ increases we clearly lose some of the mid to high level frequencies.

D.4.3 Gaussian Spaces and VAE Lipschitzness

We know that by Proposition 6.4 the variance of a function (that is once differentiable) provides a relatively tight lower bound on the Lipschitz constant of a neural network. This lower bound also increases as the variance of the Gaussian measure decreases. In the Supplement we demonstrate that this modulation of the encoder and decoder Lipschitzness is a purveyor of adversarial robustness for the VAE networks.

In previous sections we demonstrated that larger encoder variances induce smaller variance decoder functions, with lower frequency components. As the encoder variance $\boldsymbol{\sigma}^2_\phi(\mathbf{x})$ increases, the variance of the decoder function $\text{Var}_{q_\phi(\mathbf{z}|\mathbf{x})}\left(g_\theta(\mathbf{z})\right)$ decreases, which by Proposition 6.4 will lead to a smaller lower bound on the Lipschitz constant of the decoder. Due to the computational costs of estimating the Lipschitz constants of networks, we restrict our empirical analysis here to fully-connected VAEs and use *layer-wise* LipSDP (Fazlyab et al., 2019) to obtain estimates. In Table D.4.1 we confirm that regulating the encoder variance allows for us to impose a soft constraint of the Lipschitzness of the VAE decoder. We also show that adding Gaussian noise on data, as motivated in the previous sections by its effect on $\text{Var}(\mu_\phi)$, gives us finer control on the Lipschitz constant of the encoder.

Table D.4.1: The Lipschitz constants (L) of VAEs' **[left]** decoder networks (g_θ) when trained with fixed encoder scale (σ_ϕ) and **[right]** encoder networks (μ_ϕ) when trained with σ-scale Gaussian noise injection on inputs and fixed encoder variance σ_ϕ=0.5. We train fully-connected VAEs with the Sigmoid activation in their networks; on sinc($5t$) ($\dim(\mathcal{Z}) = 1$), on CelebA ($\dim(\mathcal{Z}) = 64$), and CIFAR10 ($\dim(\mathcal{Z}) = 64$). Each network has 3 dense layers each with 256 units. As σ_ϕ decreases the Lipschitz constant (L) increases, as predicted by Proposition 6.4. As σ decreases, the Lipschitz constant (L) of the encoder mean (μ_ϕ) increases. ($\pm$) is the std. dev. over 3 random seeds.

Data	$L(g_\theta; \sigma_\phi)$			$L(\mu_\phi; \sigma)$		
	$\sigma_\phi = 1.0$	$\sigma_\phi = 0.5$	$\sigma_\phi = 0.1$	$\sigma = 1.0$	$\sigma = 0.5$	$\sigma = 0.0$
sinc	2.2 ± 0.2	5.2 ± 0.3	17.9 ± 3.2	13.9 ± 2.7	24.6 ± 1.7	29.8 ± 2.2
CelebA(10^4)	7.5 ± 1.1	12.0 ± 0.5	13.7 ± 1.2	1.4 ± 0.1	1.6 ± 0.1	1.8 ± 0.1
CIFAR10(10^2)	17.9 ± 1.2	19.1 ± 1.2	27.3 ± 0.3	4.7 ± 0.2	5.6 ± 0.6	8.5 ± 0.8

Lipschitzness and Robustness Recent work shows that larger encoder variances and smaller encoder and decoder network Lipschitz constants are core contributors to the robustness of VAEs to adversarial attack (Camuto et al., 2021b; Barrett et al., 2021).

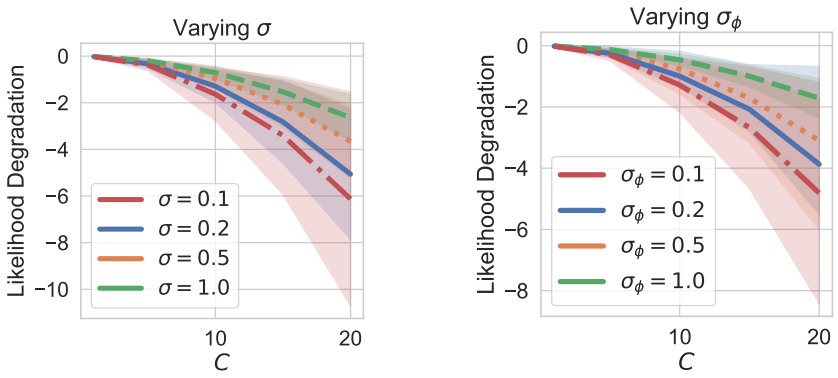


Figure D.4.3: Both plots show the relative log likelihood degradation resulting from a 'maximum-damage' (Camuto et al., 2020) adversarial attack against the maximum attack norm C. As σ, the variance of isotropic Gaussian noise added to data, and σ_ϕ, the fixed posterior variance, increase, the degradation lessens as C increases, highlighting that both these parameters improve robustness. Shading corresponds to the variance of the likelihood degradation over 25 points from CIFAR10.

Whereas this theory has viewed the network Lipschitz constants and the encoder variance as distinct parameters to control to attain robustness; our harmonic analysis shows that they are in fact intrinsically linked. This means that the encoder variance can serve as a single parameter on which we can act to improve the robustness of VAEs to adversarial attack, in that it also affects the Lipschitz constant of the decoder. Further, adding noise to the VAE input data gives us similar control on the Lipschitz constant of the encoder mean μ_ϕ. As such, fixing and modulating both the encoder variance and the variance of the noise on data allows for the imposition of soft Lipschitz constraints on the VAE networks that improve adversarial robustness.

We consider an adversary trying to distort the input data to maximally disrupt a VAE's output, as in *maximum damage* attacks (Camuto et al., 2020). The adversary maximises, with respect to a perturbation on data $\boldsymbol{\delta}_x$, constrained in norm by a constant C, the distance between the VAE reconstruction and data $\mathbf{x}$

$$\boldsymbol{\delta}_x^* = \underset{\|\boldsymbol{\delta}_x\|_2 \leq C}{\arg\max} \left(||g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}+\boldsymbol{\delta}_x) + \boldsymbol{\eta}\sigma_\phi(\mathbf{x}+\boldsymbol{\delta}_x)) - g_\theta(\boldsymbol{\mu}_\phi(\mathbf{x}))||_2 \right). \tag{D.14}$$

Given an embedding $\mathbf{z}^*$ formed from the mean encoding of $\mathbf{x}+\boldsymbol{\delta}_x$, we measure the likelihood of the original point $\mathbf{x}$ and quantify the degradation in model performance as the relative log likelihood degradation $(|\log p_\theta(\mathbf{x}|\mathbf{z}^*) - \log p_\theta(\mathbf{x}|\mathbf{z})|/\log p_\theta(\mathbf{x}|\mathbf{z}))$, where $\mathbf{z}$ is the

embedding of $\mathbf{x}$.

Figure D.4.3 shows that as the variance of noise on data (σ) and the fixed encoder variance (σ_ϕ) increase, this degradation lessens for the VAEs trained on the CIFAR10 dataset in Table D.4.1, indicating less damaging attacks. In tandem, Table D.4.1 also shows that the increase of both these parameters decreases the Lipschitz constants of the encoder and decoder network for these VAEs. Thus we confirm the analysis above (and the recent theory on the adversarial robustness of VAEs (Camuto et al., 2021b; Barrett et al., 2021)), empirically demonstrating that both σ_ϕ and σ reduce the Lipschitz constant of both the encoder and decoder networks and simultaneously improve the robustness of VAEs trained on CIFAR10.

APPENDIX E

BIBLIOGRAPHY

Jonas Adler and Sebastian Lunz. Banach Wasserstein GAN. In *NeurIPS*, 2018.

Sergios Agapiou, Andrew M. Stuart, and Yuan Xiang Zhang. Bayesian posterior contraction rates for linear severely ill-posed inverse problems. *Journal of Inverse and Ill-Posed Problems*, 22(3):297–321, 2014.

Rita Aleksziev. Tangent Space Separability in Feedforward Neural Networks. In *NeurIPS*, 2019.

Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. *ICLR*, 2017.

Raman Arora, Peter Bartlett, Poorya Mianjy, and Nathan Srebro. Dropout: Explicit Forms and Capacity Control. *arXiv:2003.03397*, 2020.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, 2019.

Sonali Bagchi and Sanjit K. Mitra. *The Nonuniform Discrete Fourier Transform and Its Applications in Signal Processing*. Kluwer Academic Publishers, USA, 1999.

Ben Barrett, Alexander Camuto, Matthew Willetts, and Tom Rainforth. Certifiably Robust Variational Autoencoders. *arXiv*, 2021.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 2013.

Chris M. Bishop. Training with Noise is Equivalent to Tikhonov Regularization. *Neural Computation*, 1995.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 2017.

Aleksandar Botev, Hippolyt Ritter, and David Barber. Practical Gauss-Newton optimisation for deep learning. *ICML*, 2017.

Martin Burger and Andreas Neubauer. Analysis of Tikhonov regularization for function approximation by neural networks. *Neural Networks*, 2003.

Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-VAE. In *Neurips*, 2018.

Alexander Camuto, Matthew Willetts, Umut Şimşekli, Stephen Roberts, and Chris Holmes. Explicit Regularisation in Gaussian Noise Injections. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Alexander Camuto, Xiaoyu Wang, Lingjiong Zhu, Chris Holmes, Mert Gürbüzbalaban, and Umut Şimşekli. Asymmetric heavy tails and implicit bias in gaussian noise injections, 2021a.

Alexander Camuto, Matthew Willetts, Stephen Roberts, Chris Holmes, and Tom Rainforth. Towards a theoretical understanding of the robustness of variational autoencoders. *AISTATS*, 2021b.

Tian Qi Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in variational autoencoders. *ICLR - Workshop Track Proceedings*, (NeurIPS), 2018.

Zixiang Chen, Yuan Cao, Quanquan Gu, and Tong Zhang. A generalized neural tangent kernel analysis for two-layer neural networks, 2020.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On Lazy Training in Differentiable Programming. In *NeurIPS*, December 2019.

Jeremy Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:2323–2356, 2019.

Taco S. Cohen and Max Welling. Group equivariant convolutional networks. *33rd International Conference on Machine Learning, ICML 2016*, 6:4375–4386, 2016.

G. M. Constantine and T. H. Savits. A multivariate faa di bruno formula with applications. *Transactions of the American Mathematical Society*, 348(2):503–520, 1996.

Felipe Cucker and Steve Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, 2002.

Wojciech Marian Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, and Razvan Pascanu. Sobolev training for neural networks. *Advances in Neural Information Processing Systems (Neurips)*, 2017.

A P Dawid. The Well-Calibrated Bayesian. *Journal of the American Statistical Association*, 1982.

Morris H. DeGroot and Stephen E. Fienberg. The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 32:12–22, 1983.

Adji B Dieng, Rajesh Ranganath, Jaan Altosaar, and David M Blei. Noisin: Unbiased regularization for recurrent neural networks. In *ICML*, 2018.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. *ICML*, 2017.

Jinqiao Duan. *An Introduction to Stochastic Dynamics*. Cambridge University Press, New York, 2015.

Alain Durmus and Eric Moulines. Nonasymptotic convergence analysis for the unadjusted Langevin algorithm. *Annals of Applied Probability*, 27(3):1551–1587, 2017.

Bartłomiej Dybiec, Ewa Gudowska-Nowak, and IM Sokolov. Stationary states in Langevin dynamics under asymmetric Lévy noises. *Physical Review E*, 76(4):041122, 2007.

Morris L. Eaton. *Group Invariance Applications in Statistics*, volume 1. 1989. ISBN 0940600153.

Sebastian Farquhar, Lewis Smith, and Yarin Gal. Try Depth Instead of Weight Correlations: Mean-field is a Less Restrictive Assumption for Deeper Networks. (NeurIPS), 2020. URL http://arxiv.org/abs/2002.03704.

Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J. Pappas. Efficient and accurate estimation of lipschitz constants for deep neural networks, 2019.

Anthony Fiche, Jean Christophe Cexus, Arnaud Martin, and Ali Khenchaf. Features modeling with an α-stable distribution: Application to pattern recognition based on continuous belief functions. *Information Fusion*, 14(4):504–520, 2013.

Jean Yves Franceschi, Alhussein Fawzi, and Omar Fawzi. Robustness of classifiers to uniform lp and gaussian noise. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1280–1288, 2018.

Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36: 193–202, 1980.

Xuefeng Gao, Mert Gürbüzbalaban, and Lingjiong Zhu. Global Convergence of Stochastic Gradient Hamiltonian Monte Carlo for Non-Convex Stochastic Optimization: Non-Asymptotic Performance Bounds and Momentum-Based Acceleration. *arXiv:1809.04618*, 2018.

Xuefeng Gao, Mert Gürbüzbalaban, and Lingjiong Zhu. Breaking reversibility accelerates Langevin dynamics for global non-convex optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Partha Ghosh, Arpan Losalka, and Michael J Black. Resisting Adversarial Attacks using Gaussian Mixture Variational Autoencoders. 2018.

Partha Ghosh, Mehdi S M Sajjadi, Antonio Vergari, Michael Black, and Bernhard Schölkopf. From Variational to Deterministic Autoencoders. In *ICLR*, 2020.

F Girosi and T Poggio. Biological Cybernetics Networks and the Best Approximation Property. *Artificial Intelligence*, 176:169–176, 1990.

Boris Vladimirovich Gnedenko and Andrey Kolmogorov. *Limit Distributions for Sums of Independent Random Variables*. Addison-Wesley, Cambridge, MA, 1954. Translated by Kai Lai Chung.

George Gondim-Ribeiro, Pedro Tabacof, and Eduardo Valle. Adversarial Attacks on Variational Autoencoders. 2018a. URL http://arxiv.org/abs/1806.04646.

George Gondim-Ribeiro, Pedro Tabacof, and Eduardo Valle. Adversarial Attacks on Variational Autoencoders. 2018b.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *ICLR*, 2015.

Rudolf Gorenflo and Francesco Mainardi. Random walk models for space-fractional diffusion processes. *Fractional Calculus & Applied Analysis*, 1:167–191, 1998.

Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael Cree. Regularisation of neural networks by enforcing lipschitz continuity. *arXiv*, 2018.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. *34th International Conference on Machine Learning, ICML 2017*, 3: 2130–2143, 2017.

Mert Gürbüzbalaban and Yuanhan Hu. Fractional moment-preserving initialization schemes for training fully-connected neural networks. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, 2021.

Mert Gürbüzbalaban, Umut Şimşekli, and Lingjiong Zhu. The heavy-tail phenomenon in SGD. In *International Conference on Machine Learning*, 2021.

Michael Hauser and Asok Ray. Principles of riemannian geometry in neural networks. *Advances in Neural Information Processing Systems*, 2017.

Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.

Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Advances in Neural Information Processing Systems*, 2017.

David P. Helmbold and Philip M. Long. On the inductive bias of dropout. *Journal of Machine Learning Research*, 16:3403–3454, 2015.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. 2017a.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017b.

Irina Higgins, Arka Pal, Andrei Rusu, Loie Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Lexander Lerchner. DARLA: Improving zero-shot transfer in reinforcement learning. In *ICML*, 2017c.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design. *ICML*, 2019.

Liam Hodgkinson and Michael W Mahoney. Multiplicative noise and heavy tails in stochastic optimization. *arXiv preprint arXiv:2006.06293*, 2020.

C. C. Holmes and S. G. Walker. Assigning a value to a power likelihood in a general Bayesian model. *Biometrika*, 2017.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 1991.

Peter Imkeller and Ilya Pavlyukevich. First exit times of SDEs driven by stable Lévy processes. *Stochastic Processes and their Applications*, 116(4):611–642, 2006.

Peter Imkeller and Ilya Pavlyukevich. Metastable behavior of small noise Lévy-driven diffusions. *ESAIM: Probability and Statistics*, 12:412–437, 2008.

Peter Imkeller, Ilya Pavlyukevich, and Michael Stauch. First Exit Times of Non-linear Dynamical Systems in &R;^{d} Perturbed by Multifractal Lévy Noise. *Journal of Statistical Physics*, 141(1), 2010.

Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *NeurIPS*, 2018.

Daniel Jakubovitz and Raja Giryes. Improving DNN robustness to adversarial attacks using Jacobian regularization. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018.

Svante Janson. *Gaussian Hilbert Spaces*. Cambridge Tracts in Mathematics. Cambridge University Press, 1997.

Stanisław Jastrzębski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three Factors Influencing Minima in SGD. *arXiv:1711.04623*, 2017.

Oliver Johnson and Richard Samworth. Central limit theorem and convergence to stable laws in Mallows distance. *Bernoulli*, 11(5):829–845, 2005.

Nitish Shirish Keskar, Jorge Nocedal, Ping Tak Peter Tang, Dheevatsa Mudigere, and Mikhail Smelyanskiy. On large-batch training for deep learning: Generalization gap and sharp minima. *ICLR*, 2019.

Diederik P Kingma and Jimmy Lei Ba. Adam: A Method for Stochastic Optimisation. In *ICLR*, 2015.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2014.

Diederik P. Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, 2015.

J Kos, I Fischer, and D Song. Adversarial Examples for Generative Models. In *IEEE Security and Privacy Workshops*, 2018a.

Jernej Kos, Ian Fischer, and Dawn Song. Adversarial examples for generative models. *IEEE Symposium on Security and Privacy Workshops*, pages 36–42, 2018b. doi: 10.1109/SPW.2018.00014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (Neurips)*, 2012.

M.J. Kroneburg. The binomial coefficient for negative arguments. *arXiv preprint arXiv:1105.3689*, 2011.

Arun Kumar Kuchibhotla and Abhishek Chakrabortty. Moving beyond sub-Gaussianity in high dimensional statistics: Applications in covariance estimation and linear regression. *arXiv:1804.02605*, 2018.

S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 1951.

Abhishek Kumar and Ben Poole. On Implicit Regularization in β-VAEs. *ICML*, 2020.

Daniel Kunin, Jonathan M. Bloom, Aleksandrina Goeva, and Cotton Seed. Loss landscapes of regularized linear autoencoders. *ICML*, 2019.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1, 1989.

Yann LeCun, Leon Bottou, Genevieve B. Orr, and Klaus Robert Müller. Efficient BackProp. 1998a.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998b. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.7665.

Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. *Proceedings - IEEE Symposium on Security and Privacy*, 2019.

Bai Li, Changyou Chen, Wenlin Wang, and Lawrence Carin. Certified Adversarial Robustness with Additive Noise. In *NeurIPS*, 2018.

Qianxiao Li, Cheng Tai, and Weinan E. Stochastic modified equations and adaptive stochastic gradient algorithms. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2101–2110, 06–11 Aug 2017.

Yuhang Liu, Wenyong Dong, Lei Zhang, Dong Gong, and Qinfeng Shi. Variational bayesian dropout with a hierarchical prior. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019.

Francesco Locatello, Stefan Bauer, Mario Lucie, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *ICML*, 2019.

Francesco Mainardi, Yuri Luchko, and Gianni Pagnini. The fundamental solution of the space-time fractional diffusion equation. *Fractional Calculus & Applied Analysis*, 4: 153–192, 2001.

Stephan Mandt, Matthew D. Hoffman, and David M. Blei. A variational analysis of stochastic gradient algorithms. *33rd International Conference on Machine Learning, ICML 2016*, 1:555–566, 2016.

Emile Mathieu, Tom Rainforth, N. Siddharth, and Yee Whye Teh. Disentangling disentanglement in variational autoencoders. In *ICML*, 2019.

Mark Meerschaert and Charles Tadjeran. Finite difference approximation for fractional advection-dispersion flow equations. *Journal of Computational and Applied Mathematics*, 172:65–77, 2004.

Barbara Mele and Guido Altarelli. Lepton spectra as a measure of b quark polarization at LEP. *Physics Letters B*, 299(3-4):345–350, 1993.

Jeffrey W Miller and David B Dunson. Robust bayesian inference via coarsening. *Journal of the American Statistical Association*, 114(527):1113–1125, 2019.

Seyed Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

Saralees Nadarajah and Tibor K Pogány. On the distribution of the product of correlated normal random variables. *Comptes Rendus Mathematique*, 354(2):201–204, 2016.

Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using Bayesian Binning. *Proceedings of the National Conference on Artificial Intelligence*, 4:2901–2907, 2015. ISSN 2159-5399.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *arXiv preprint arXiv:1912.02292*, 2019.

Eric Nalisnick and Padhraic Smyth. Learning priors for invariance. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 84, pages 366–375, 2018.

J. Neyman and E. S. Pearson. On the Problem of the Most Efficient Tests of Statistical Hypotheses. 1933.

Behnam Neyshabur and Nathan Srebro. Norm-Based Capacity Control in Neural Networks.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro. Exploring generalization in deep learning. *Advances in Neural Information Processing Systems (Neurips)*, 2017.

Andre T Nguyen and Edward Raff. Adversarial Attacks, Regression, and Numerical Stability Regularization. In *The AAAI Workshop on Engineering Dependable and Secure Machine Learning Systems*, 2019.

Thanh Huy Nguyen, Umut Şimşekli, Mert Gürbüzbalaban, and Gaël Richard. First Exit Time Analysis of Stochastic Gradient Descent Under Heavy-Tailed Gradient Noise. In *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, Vancouver, Canada, 2019.

Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. *ICML*, 2005.

John P. Nolan. Maximum likelihood estimation and diagnostics for stable distributions. In Ole E. Barndorff-Nielsen, Sidney I. Resnick, and Thomas Mikosch, editors, *Lévy Processes: Theory and Applications*, pages 379–400. Birkhäuser Boston, Boston, MA, 2001.

Erich Novak, Mario Ullrich, Henryk Woźniakowski, and Shun Zhang. Reproducing kernels of Sobolev spaces on d and applications to embedding constants and tractability. *Analysis and Applications*, 16(5):693–715, 2018.

Amilcar Oliveira, Teresa A. Oliveira, and Antonio Seijas-Macias. Skewness into the product of two normally distributed variables and the risk consequences. *Revstat Statistical Journal*, 14(2):119–138, 2016.

Manuel Duarte Ortigueira. Riesz potential operators and inverses via fractional centred derivatives. *International Journal of Mathematics and Mathematical Sciences*, 2006 (Article ID 48391):1–12, 2006.

Manuel Duarte Ortiguera. Fractional central differences and derivatives. *IFAC Proceedings Volumes*, 39(11):58–63, 2006b.

Fabien Panloup. Recursive computation of the invariant measure of a stochastic differential equation driven by a Lévy process. *Annals of Applied Probability*, 18(2):379–426, 2008.

Nicolas Papernot, Patrick Mcdaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. *Proceedings - 2016 IEEE European Symposium on Security and Privacy, EURO S and P 2016*, pages 372–387, 2016. doi: 10.1109/EuroSP.2016.36.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Ben Poole, Jascha Sohl-Dickstein, and Surya Ganguli. Analyzing noise in autoencoders and deep networks. *arXiv:1406.1831*, 2014.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. *Advances in Neural Information Processing Systems*, pages 3368–3376, 2016. ISSN 10495258.

Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.

Maxim Raginsky, Alexander Rakhlin, and Matus Telgarsky. Non-convex learning via stochastic gradient Langevin dynamics: a nonasymptotic analysis. In *Conference on Learning Theory*, pages 1674–1703, 2017.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A Hamprecht, Yoshua Bengio, and Aaron Courville. On the Spectral Bias of Neural Networks. (1), 2019.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014a.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *ICML*, 2014b.

Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

Gareth O Roberts and Osnat Stramer. Langevin diffusions and Metropolis-Hastings algorithms. *Methodology and Computing in Applied Probability*, 4(4):337–357, 2002.

Michal Rolinek, Dominik Zietlow, and Georg Martius. Variational autoencoders pursue PCA directions (by accident. In *IEEE CVPR*, 2019.

Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. *American Journal of Psychology*, 76:705, 1963.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv:1609.04747*, 2016.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*. 1986.

Levent Sagun, Utku Evci, V. Ugur Güney, Yann Dauphin, and Léon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *ICLR - Workshop Track Proceedings*, 2018.

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. PixelCNN++: Improving the PixelCnn with discretized logistic mixture likelihood and other modifications. In *ICLR*, 2017.

Gennady Samorodnitsky and Murad S. Taqqu. *Stable Non-Gaussian Random Processes: Stochastic Models with Infinite Variance*. 1994.

Katayoon Sarafrazi and Mehran Yazdi. Skewed alpha-stable distribution for natural texture modeling and segmentation in contourlet domain. *Eurasip Journal on Image and Video Processing*, 2019.

Daniel Schertzer, Michele Larchevêque, Jinqiao Duan, Vladimir Yanovsky, and Shaun Lovejoy. Fractional Fokker-Planck equation for nonlinear stochastic differential equations driven by non-Gaussian Lévy stable noises. *Journal of Mathematical Physics*, 42 (1):200–212, 2001.

Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on MnIsT. *ICLR*, 2019.

Adi Shamir, Itay Safran, Eyal Ronen, and Orr Dunkelman. A Simple Explanation for the Existence of Adversarial Examples with Small Hamming Distance. 2019. URL http://arxiv.org/abs/1901.10861.

Rui Shu, Hung H. Bui, Shengjia Zhao, Mykel J. Kochenderfer, and Stefano Ermon. Amortized inference regularization. In *NeurIPS*, 2018.

Umut Şimşekli. Fractional Langevin Monte Carlo: Exploring Lévy driven stochastic differential equations for Markov Chain Monte Carlo. In *International Conference on Machine Learning*, pages 3200–3209, 2017.

Umut Şimşekli, Levent Sagun, and Mert Gürbüzbalaban. A tail-index analysis of stochastic gradient noise in deep neural networks. In *ICML*, 2019.

Umut Şimşekli, Lingjiong Zhu, Yee Whye Teh, and Mert Gürbüzbalaban. Fractional Underdamped Langevin Dynamics: Retargeting SGD with Momentum under Heavy-Tailed Gradient Noise. In *ICML*, 2020.

Umut Şimşekli, Mert Gürbüzbalaban, Thanh Huy Nguyen, Gaël Richard, and Levent Sagun. On the Heavy-Tailed Theory of Stochastic Gradient Descent for Deep Neural Networks. *arXiv:1912.00018*, 2019a.

Umut Şimşekli, Levent Sagun, and Mert Giirbiizbalaban. A tail-index analysis of stochastic gradient noise in deep neural networks. *ICML*, 2019b.

O Yu Sliusarenko, DA Surkov, V Yu Gonchar, and Aleksei V Chechkin. Stationary states in bistable system driven by Lévy noise. *The European Physical Journal Special Topics*, 2013.

Jure Sokolić, Raja Giryes, Guillermo Sapiro, and Miguel R.D. Rodrigues. Robust Large Margin Deep Neural Networks. *IEEE Transactions on Signal Processing*, 2017.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *ICLR*, 2014.

Pedro Tabacof, Julia Tavares, and Eduardo Valle. Adversarial Images for Variational Autoencoders. 2016. URL http://arxiv.org/abs/1612.00155.

Dávid Terjék. Adversarial lipschitz regularization, 2020.

Wenyi Tian, Han Zhou, and Weihua Deng. A class of second order difference approximations for solving space fractional diffusion equations. *Mathematics of Computation*, 2015.

A N Tikhonov and V I A Arsenin. *Solutions of ill-posed problems*. Scripta series in mathematics. 1977.

Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. In *NeuRIPS*, 2018.

Amos Tversky and Daniel Kahneman. Judgment under uncertainty: Heuristics and biases. *Science*, 1974.

Jonathan Uesato, Brendan O'Donoghue, Aaron van den Oord, and Pushmeet Kohli. Adversarial risk and the dangers of evaluating against weak attacks. *arXiv*, 2018.

Roman Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2018.

Mariia Vladimirova, Jakob Verbeek, Pablo Mesejo, and Julyan Arbel. Understanding priors in Bayesian neural networks at the unit level. *ICML*, 2019.

Mariia Vladimirova, Stéphane Girard, Hien Nguyen, and Julyan Arbel. Sub-Weibull distributions: generalizing sub-Gaussian and sub-Exponential properties to heavier-tailed distributions. *Stat*, 2020.

Stefan Wager, Sida Wang, and Percy S Liang. Dropout training as adaptive regularization. In *Advances in neural information processing systems (Neurips)*, 2013.

G. Wahba. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics, 1990.

Sida I. Wang and Christopher D. Manning. Fast dropout training. In *ICML*, 2013.

Andrew R. Webb. Functional Approximation by Feed-Forward Networks: A Least-Squares Approach to Generalization. *IEEE Transactions on Neural Networks*, 1994.

Colin Wei, Sham Kakade, and Tengyu Ma. The Implicit and Explicit Regularization Effects of Dropout. In *ICML*, 2020.

Max Welling and Yee Whye Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *ICML*, 2011.

Florian Wenzel, Kevin Roth, Bastiaan S. Veeling, Jakub Świątkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How Good is the Bayes Posterior in Deep Neural Networks Really? 2020.

Matthew Willetts, Alexander Camuto, Stephen Roberts, and Chris Holmes. Regularising Deep Networks with Deep Generative Models. *arXiv:1909.11507*, 2019.

Matthew Willetts, Alexander Camuto, Tom Rainforth, Stephen Roberts, and Chris Holmes. Improving VAEs' Robustness to Adversarial Attack. In *ICLR*, 2020.

Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Ruslan Salakhutdinov, and Kamalika Chaudhuri. Adversarial robustness through local lipschitzness. *arXiv*, 2020.

Dong Yin, Raphael Gontijo Lopes, Jon Shlens, Ekin Dogus Cubuk, and Justin Gilmer. A fourier perspective on model robustness in computer vision. In *NeurIPS*, 2019.

Chiyuan Zhang, Benjamin Recht, Samy Bengio, Moritz Hardt, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models? In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020a.

Mingtian Zhang, Peter Hayes, Thomas Bird, Raza Habib, and David Barber. Spread Divergence. *ICML*, 2020b.

Pan Zhou, Jiashi Feng, Chao Ma, Caiming Xiong, Steven Hoi, and Weinan E. Towards theoretically understanding why SGD generalizes better than ADAM in deep learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.