

UCLA

UCLA Electronic Theses and Dissertations

Title

Part I: The geometry and manipulation of natural data for optimizing neural networks Part II:
A theory for undercompressive shocks in tears of wine

Permalink

<https://escholarship.org/uc/item/7w0278f5>

Author

Dukler, Yonatan

Publication Date

2021

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Part I: The geometry and manipulation of natural data
for optimizing neural networks

Part II: A theory for undercompressive shocks in tears of wine

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Mathematics

by

Yonatan Dukler

2021

© Copyright by
Yonatan Dukler
2021

ABSTRACT OF THE DISSERTATION

Part I: The geometry and manipulation of natural data
for optimizing neural networks

Part II: A theory for undercompressive shocks in tears of wine

by

Yonatan Dukler

Doctor of Philosophy in Mathematics

University of California, Los Angeles, 2021

Professor Andrea Bertozzi, Co-Chair

Professor Guido Francisco Montúfar Cuartas, Co-Chair

Abstract: In Part I of the thesis, we present a body of work analyzing and deriving data-centric regularization methods for the effective training of machine learning models. Machine learning and deep learning in particular have been highly successful in computer vision and generative modelling in recent years. Nonetheless, the progress of such approaches crucially relies on effective regularization, architectural, and algorithmic choices that are often abstracted away during a first consideration. In this part we present the reader with effective regularization approaches focused on the geometry and biases of natural data and parameterization of deep neural networks. We start by deriving a regularization to accurately capture geometric robustness and natural variances of images in Chapter 1. This approach enables significant improvement in model robustness and relies on the theory of optimal transport which we introduce alongside with our method in the chapter. Dataset regularization is extended to active manipulation of the sampling distribution as opposed to each datum in Chapter 2. In the chapter, we present a general and differentiable technique for dataset

optimization enabling debiasing of noisy and imbalanced datasets. In our final contribution for Part I, In Chapter 3, we study the interplay between data and model parameterization. This concerns with the widely-spread architectural approach of neural network normalization. We analyze the convergence dynamics of Weight Normalization and present the first proof of global convergence for dynamically normalized ReLU networks when trained with gradient descent.

In Part II, we study the fluid dynamics phenomena known as the tears of wine problem for thin films in water-ethanol mixtures and present a model for the climbing dynamics. The new formulation includes a Marangoni stress balanced by both the normal and tangential components of gravity as well as surface tension which lead to distinctly different behavior. The prior literature did not address the wine tears but rather the behavior of the film at earlier stages and the behavior of the meniscus. In the lubrication limit we obtain an equation that is already well-known for rising films in the presence of thermal gradients. Such models can exhibit nonclassical shocks that are undercompressive. We present basic theory that allows one to identify the signature of an undercompressive wave. We observe both compressive and undercompressive waves in new experiments and we argue that, in the case of a preswirled glass, the famous “wine tears” emerge from a reverse undercompressive shock originating at the meniscus.

The dissertation of Yonatan Dukler is approved.

Stanley J. Osher

Quanquan Gu

Guido Francisco Montúfar Cuartas, Committee Co-Chair

Andrea Bertozzi, Committee Co-Chair

University of California, Los Angeles

2021

TABLE OF CONTENTS

I The geometry and manipulation of natural data for optimizing neural networks **1**

Overview for Part I **2**

1 Natural image prior and regularization of vision tasks via the Wasserstein metric **5**

1.1 Introduction 5

1.2 Mathematics of optimal transport and the Wasserstein Ground Metric 8

1.2.1 Wasserstein-1 metric 9

1.2.2 Wasserstein-2 metric 10

1.2.3 Wasserstein metric on graphs 11

1.2.4 Riemannian calculus of \mathcal{W}^2 11

1.2.5 Wasserstein-2 gradient on discrete sample space 13

1.2.6 Efficient implementation of the Wasserstein gradient norm 16

1.3 Related works 17

1.4 Wasserstein of Wasserstein loss for learning generative models 19

1.4.1 Wasserstein of Wasserstein loss 21

1.4.2 Relevant literature for the Wasserstein of Wasserstein loss 27

1.4.3 Wasserstein of Wasserstein GANs 27

1.4.4 Experiments 31

1.4.5 Discussion 37

1.5 Wasserstein Tikhonov regularization in image classification 37

1.5.1	Introduction	37
1.5.2	Relevant literature to Wasserstein adversarial robustness	40
1.5.3	Adversarial training and ground truth geometry	41
1.5.4	Perturbed loss and Wasserstein diffusion Tikhonov regularizer	43
1.5.5	Experiments	46
1.5.6	Discussion	48
1.A	Appendix	50
1.A.1	Proof of equivalence of noise training with Wasserstein Thikonov Regularization	50
1.A.2	Wasserstein metric in un-normalized distributions	51
1.A.3	Detailed description of the experiments	52
1.A.4	WWGAN generated images	54
2	Differentiable dataset optimization	57
2.1	Introduction	57
2.2	Related work	59
2.3	Proposed method	62
2.3.1	Linearization	64
2.3.2	Computation of the dataset derivative	65
2.3.3	Leave-one-out optimization	67
2.3.4	Dataset optimization with DIVA	68
2.4	Experimental results	69
2.5	Discussion	74
2.A	Appendix	75

2.A.1	Additional experiments	75
2.A.2	Experimental details	77
2.A.3	Proofs of propositions	79
3	On the dynamics and convergence of Weight Normalization for training neural networks	86
3.1	Introduction	86
3.2	Related work	89
3.3	Weight Normalization	91
3.4	Evolution dynamics	94
3.5	Main convergence theory	96
3.6	Discussion	103
3.A	Appendix	105
3.A.1	Weight Normalization dynamics proofs	106
3.A.2	Convergence proof for gradient flow	107
3.A.3	Convergence proof for finite step-size training	129
	Final remarks for Part I	146
II	A theory for undercompressive shocks in tears of wine	149
4	Modelling the tears of wine phenomena	151
4.1	Introduction	151
4.2	Hydrodynamic model	153
4.3	Meniscus-driven film climbing and nonclassical shocks	158

4.4	Experimental survey and simulations	163
4.5	Conical shaped substrate	167
4.6	Reverse undercompressive shocks on a preswirled substrate	169
4.7	Conclusion	176
4.A	Appendix: Extended survey of prior experimental works	178
	References	182

LIST OF FIGURES

1.1	Fixed distance Wasserstein and Euclidean interpolates	7
1.2	CIFAR-10 nearest neighbors L2, Wasserstein metrics	21
1.3	Illustration of Wasserstein- p loss function with Wasserstein- q ground metric.	23
1.4	Wasserstein of Wasserstein loss visualization	25
1.5	Discriminator robustness to image translation	34
1.6	Robustness to Salt and pepper perturbations	36
1.7	FID curves for WGAN-GP, WWGAN	36
1.8	Translation augmentation illustration	48
1.9	CelebA cropped 64×64 WWGAN generated images.	55
1.10	CIFAR-10 WWGAN generated images.	56
2.1	Diagram of differentiable bi-level optimization	64
2.2	Examples of the reweighting done by DIVA.	67
2.3	Outlier detection using DIVA	72
2.4	DIVA Extend incremental sample addition	72
2.5	Augmentation testing using DIVA	74
2.6	Test and validation error as a function of dataset-reweighting step-size	76
2.7	DIVA Extend incremental sample addition plot (all datasets)	76
3.1	Schematic of (continuous and discrete) weight updates of weight-normalized network parameters	93
4.1	Schematic illustration of tears of wine climbing	154
4.2	Visualizing shock characteristics	156

4.3	Numerical simulation of PDE (4.7)	159
4.4	Shock bifurcation diagram	162
4.5	Numerical simulations of Vuiellemuier <i>et al.</i> experiments (BI), (BII) for long times.	164
4.6	A comparison of shock types affected by the precursor thickness b	165
4.7	The evolution of film height of [VS15] (wine setting) with $b = 0.028$	165
4.8	Spontaneous climb images of ethanol-water mixture with ethanol concentration $C = 0.7$	166
4.9	Long-time shock profiles of (4.22) for varying A	168
4.10	Top view images of tears of wine experiment at different times	169
4.11	Tears of wine experiment in conical wine glass.	170
4.12	The formation of a reverse-undercompressive (RUC) shock with initial condition (4.23) and boundary conditions (4.24)	172
4.13	Critical thickness (h_∞) for RUC shocks	174
4.14	Curvature effects in the meniscus boundary condition setting of advancing waves	174

LIST OF TABLES

1.1	Discriminator robustness to vertical translations on CIFAR-10	35
1.2	Robust test error for models trained with Wasserstein Diffusion	47
1.3	Average number of prediction flips on sequences of translated test images from CIFAR-10.	48
2.1	Test error of DIVA Reweigh on fine-grained classification	70
2.2	Test error of DIVA Extend on fine-grained classification	71
4.1	Relevant dimensional groups used in Table 4.2	179
4.2	Experimental results from literature and corresponding theory	180
4.3	Additional experimental results from literature and corresponding theory	181

ACKNOWLEDGMENTS

I would like to take this opportunity to express my gratitude to the people that made the completion of this dissertation possible and the journey of my PhD especially meaningful. I apologize in advance that this will constitute an incomplete attempt.

First, I would like to thank my advisors Andrea and Guido for their devoted and steady support. They have both taken me under their wing and guided me throughout this journey, always putting my interests first as they mentored me throughout the years of my PhD.

I encountered Andrea as an undergraduate in her ML course in 2016. At the time, I was studying analysis as a pure math student. I would like to thank Andrea for bringing me to the “dark side” ☺ of applied math from pure math. Throughout my PhD, Andrea has been there, ready to meet, share advice, and do all in her ability to support me and my research.

I met Guido, during my first year of PhD as he introduced me to my passion of deep learning and the mathematics of artificial intelligence. I could not have asked for a more knowledgeable, patient, and kind advisor than Guido. Guido has showed me the ropes of deep learning, experimentation and writing. I am indebted to Guido for his open mindedness, and generosity as my advisor.

I would like to also thank my other members of my thesis committee, Quanquan Gu and Stanley Osher for their suggestions, time, and helpful feedback. Your mentorship has greatly improved my research.

Further, I would like to also thank mentors that completely transformed my PhD experience through widening my perspectives and working on exciting projects. I would like to express my gratitude to the Custom Labels team of AWS research: Alessandro Achille, Giovanni Paolini, Avinash Ravichandran and Stefano Soatto, for their commitment to research and teaching me so much in a short amount of time. More recently I would like to thank Sergey Tulyakov for his determination and generosity as a research mentor and teaching me the importance of persistence in research. I would like to thank Quanquan Gu for sharing with me

some of his expertise in deep learning theory and would also like to thank Mihai Cucuringu and the team at the Alan Turing Institute for a great research opportunity and stay at ATI.

I would like to acknowledge the support of the NSF Graduate Research Fellowship, grant DGE-1650604 that has been an important steppingstone in my research career.

I would like to also extend big thanks to my teachers, collaborators, and fellow researchers that made my PhD more enjoyable: Professor Chris Anderson, Professor Tao Gao, Professor Johnathan Kao, Professor Deanna Needell, Professor Marc Potters; Claudia Falcon, Hangjie Ji, Wuchen Li, Alex Lin; Ben Bowman, Spencer Frei, Robert Hannah, Howard Heaton, Hui Jin, Elan Markowitz, Michael Murray, Kevin Miller, Aliaksandr Siarohin, Chris Strohmeier, Thomas Tu, and Baichuan Yuan.

Finally I will never be able to fully express my gratitude to my parents Michal and Avinoam who have never stopped encouraging me and helping me, and made the person I am today.

FUNDING ACKNOWLEDGMENTS

Below I detail the funding sources used for the research presented in this thesis.

- I was supported by the NSF GRFP grant DGE-1650604 – at least in part – in the research that is the basis for all of the chapters presented in this thesis.
- The works in Chapters 1, 3 also received support from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement no 757983).
- For the work in Chapter 1 co-author Alex Tong Lin was supported by AFOSR MURI FA9550-18-1-050.
- Part of the research in Chapter 1 was performed at the Institute for Pure and Applied Mathematics (IPAM), which is supported by the National Science Foundation (Grant No. DMS-1440415).

- The work in Chapter 2 was conducted during a paid internship at Amazon Web Services, and was supported by Amazon Inc.
- For the work in Chapter 3 co-author Quanquan Gu was supported in part by NSF CAREER Award IIS-1906169, BIGDATA IIS-1855099, and Salesforce Deep Learning Research Award.
- The research presented in Chapter 4 was partly supported by Simons Foundation Math+X investigator award number 510776.

VITA

2014 – 2017 B.S. (Mathematics), UCLA

2014 – 2017 M.A. (Mathematics), *Departmental Scholar*, UCLA

06/16 – 08/16 Undergraduate Researcher, UCLA Applied Math REU

2017 – Present Graduate Researcher, Mathematics Department, UCLA

2018 – 2021 NSF Graduate Research Fellowship

06/18 – 08/18 Visiting Researcher, Alan Turing Institute; *Host: Mihai Cucuringu*

01/20 – 04/20 Computer Vision Intern, Samsung SSIC, Advanced Technology Group

01/21 – 04/21 Research Intern, AWS Computer Vision, Amazon Inc.

06/21 – 10/21 Research Intern, Snap Research (Creative Vision), Snap Inc.

PUBLICATIONS

Y. Dukler, W. Li, A. Lin, G. Montufar, “Wasserstein of Wasserstein Loss for Learning Generative Models.” *Proceedings of the 36th International Conference on Machine Learning* PMLR 97:1716-1725, 2019.

A. Lin, Y. Dukler, W. Li, G. Montufar, “Wasserstein Diffusion Tikhonov Regularization.” *arXiv preprint arXiv:1909.06860 (2019)*. Presented in NeurIPS 2019, OTML workshop.

Y. Dukler, H. Ji, C. Falcon, and A. L. Bertozzi “Theory for undercompressive shocks in tears of wine” *Phys. Rev. Fluids* vol. 5, 034002 – Published 17 March 2020.

Y. Dukler, Q. Gu, G. Montufar, “Optimization Theory for ReLU Neural Networks Trained with Normalization Layers.” *Proceedings of the 37th International Conference on Machine Learning* PMLR 119:2751-2760, 2020.

Part I

**The geometry and manipulation of
natural data
for optimizing neural networks**

Overview for Part I

A summary of the chapters and contributions of Part I of the thesis, are given below.

- In Chapter 1 we utilize the theory of optimal transport to derive a tractable algorithm of regularizing image data using exact notions of the Wasserstein metric. Our new view on image-centric problems maps high-dimensional image data to a discrete distribution over the pixel space and its underlying geometry. This distribution based representation for each image datum enables defining an optimal transport metric distance between a pair of images, which we name the Wasserstein Ground Metric (WGM). The distribution representation of images draws a stark comparison from traditional image data representations as the WGM is shown to align with the natural variances of data in the visual modality. In particular, with the WGM local geometric perturbations, including translations and rotations are proportional to their magnitude. The framework is applied to both generative and discriminate computer vision tasks and significantly improves robustness to natural variations (e.g. translations). The new formulation utilizes the Riemannian structure of the Wasserstein-2 metric for deriving a gradient penalty on the WGM metric for discrete spaces. This computation is made highly efficient using clever use of convolution operators which are highly optimized.

The presented work in Chapter 1 is derived from the research works:

- “*Wasserstein of Wasserstein Loss for Learning Generative Models*”. which was presented at ICML 2019. This work is a collaborative project with Alex Tong-Lin, Wuchen Li, and Guido Montúfar.
 - “*Wasserstein diffusion Tikhonov regularization*”. which was presented at NeurIPS 2019 OTML workshop. This work continued the collaboration with Alex Tong-Lin, Wuchen Li, and Guido Montúfar.
- In Chapter 2 we present dataset manipulation as a differentiable procedure by deriving

the first differentiable dataset optimization approach based on the Leave One Out Error, optimized based on the final model loss. This new method extends the optimization of models to also include the importance and selection of each data sample. If done carelessly, dataset optimization leads to trivial solutions to the learning problem — thereby bypassing learning meaningful representations. For this reason, we follow the AutoML approach and present dataset optimization as a bi-level optimization procedure. Unlike other approaches for dataset optimization, we are capable of optimizing the dataset end-to-end as we derive a closed-form derivative for the the final validation loss with respect to the weight of each sample. Our method enables better dataset auto-curation tasks such as outlier rejection, dataset extension, and automatic aggregation of multi-modal data leading to consistent and significant improved model accuracy. This is all while using the same model classes that yet are trained under modified datasets. Chapter 2 corresponds to research I conducted while interning at Amazon Web Services (AWS) in 2021.

- This research project is a collaboration with Alessandro Achille, Giovanni Paolini, Avinash Ravichandran, Marzia Polito, and Stefano Soatto from the AWS research team.
- In Chapter 3 we analyze mathematically the widely-spread architectural approach of neural network normalization. This includes the normalization techniques of Batch Normalization, Weight Normalization, and Layer Normalization which apply normalization to the intermediate representations or parameters of the network during training. In the case of Weight Normalization, we prove for the first time that dynamically normalized ReLU neural networks, one of the most common deep learning architectural choices, converge to global minima when trained with gradient descent. Our analysis showcases the interaction between two parameter classes and their effects on the convergence speed. The derivation follows a contemporary body of works studying non-convex optimization of neural networks using the notion of the Neural Tangent Kernel. Overall

the discovered dynamics highlight the differences in the geometry of the optimization landscape as compared with traditional un-normalized networks. In particular we derive a modified NTK in the case of Weight Normalization that is composed of 2 kernel pieces. The 2 pieces of the NTK then describe modified dynamics under different initial conditions.

The theory work presented in Chapter 3 corresponds to the research

- “*Optimization Theory for ReLU Neural Networks Trained with Normalization Layers*”. Presented at ICML 2020. This work has been a collaborative project with Quanquan Gu and Guido Montúfar.

CHAPTER 1

Natural image prior and regularization of vision tasks via the Wasserstein Ground Metric*

1.1 Introduction

Current machine learning methods including deep learning are highly sensitive to the geometry of the data, and are often trained with massive datasets that are further extended using data augmentations to impose such geometry. In many tasks, for the model to generalize, it is of eminent importance that the model behaves well with respect to the natural variances of the samples. For example, for image based data, affine transformations such as translations, rotations, and other local stretches of the image, should not alter the perceptual content of the image. Despite the largely hand-off approach of learning from data, priors on the task are crucial. In deep learning the type of priors vary and cater to the data modality, presenting themselves in the form of architectural choices, augmentations, labelling types, and loss functions. In this chapter we present a general and lightweight regularization framework that naturally guides deep learning models based on a strong prior of the data distribution in image space. The core of the method is in representing each image datum as a mass distribution over its discrete pixel space, this allows us to endow the input space of the model with an optimal transport distance that naturally aligns with human perception and geometrical manipulation of the data. The method we present bypasses the computationally intractable cost of computing the Wasserstein metric in high dimensions (e.g. the pixel space)

*This chapter is adapted from [DLL19b, LDL19]

and results in an efficient regularization term.

Motivation for a Wasserstein image metric In modern machine learning, it is common to regularize models capacity and “smoothness” by penalizing the gradient of the model with respect to the data input. Indeed for a parameterized model f we have the linearization,

$$f(\mathbf{z}) = f(\mathbf{x}) + \nabla_{\mathbf{x}}f(\mathbf{x}) \cdot (\mathbf{z} - \mathbf{x}) + o((\mathbf{z} - \mathbf{x})^2).$$

Re-arranging and applying Cauchy-Schwarz, implies that

$$\|f(\mathbf{z}) - f(\mathbf{x})\| \leq \|\mathbf{z} - \mathbf{x}\|_2 \cdot \|\nabla_{\mathbf{x}}\mathbf{f}\|_2.$$

Therefore $\|\nabla_{\mathbf{x}}f\|_2$ may serve as an estimate of the Lipschitz constant of f , quantitatively describing the “smoothness” of f . For this reason this gradient penalty is crucially used in many deep learning frameworks for regularization [Bis95, FOA19] and enforcing continuity requirements [GAA17, ACB17, PFL17]. In fact, for linear models adding a gradient penalty term amounts to the common weight-decay regularization. Nonetheless, in the above linearization the distance between two sample images is taken to be the mean square difference over the input features, i.e., the L2 (Euclidean) norm. This, however, does not incorporate additional knowledge that we have about the space of natural images, leading to “isotropic smoothness” along arbitrary directions of the image equally. For example, the distance of translating the image 1 pixel step to the left is akin to the distance of moving each pixel in the image to any arbitrary chosen locations in the image.

Instead we suggest the Wasserstein distance over the space of images defined as histograms over pixels, having a ground metric over pixel locations. With the Wasserstein Ground Metric, repeating the same motivating example illustrates that the distance of the single pixel step is proportional to the distance on the pixel grid that the pixels are being moved. With this a Wasserstein metric on the space of images is continuous and roughly linear with respect to the magnitude of natural variances such as translations, rotations, and other local stretches. In Figure 1.1 we present the Wasserstein Ground Metric level-sets and illustrate its strength

in aligning with human perception as opposed to the L^2 distance. Therefore one would like to replace the Euclidean distance $d_{L^2}(\mathbf{x}, \mathbf{z}) = \|\mathbf{x} - \mathbf{z}\|_2$ with the Wasserstein Ground Metric $d_{W^2}(\mathbf{x}, \mathbf{z})$ and compute the corresponding linearization and gradient penalty $\|\nabla_{\mathbf{x}}^{W^2} \mathbf{f}\|_2$. In this chapter we show that this is possible to compute exactly and illustrate the benefits of this framework.

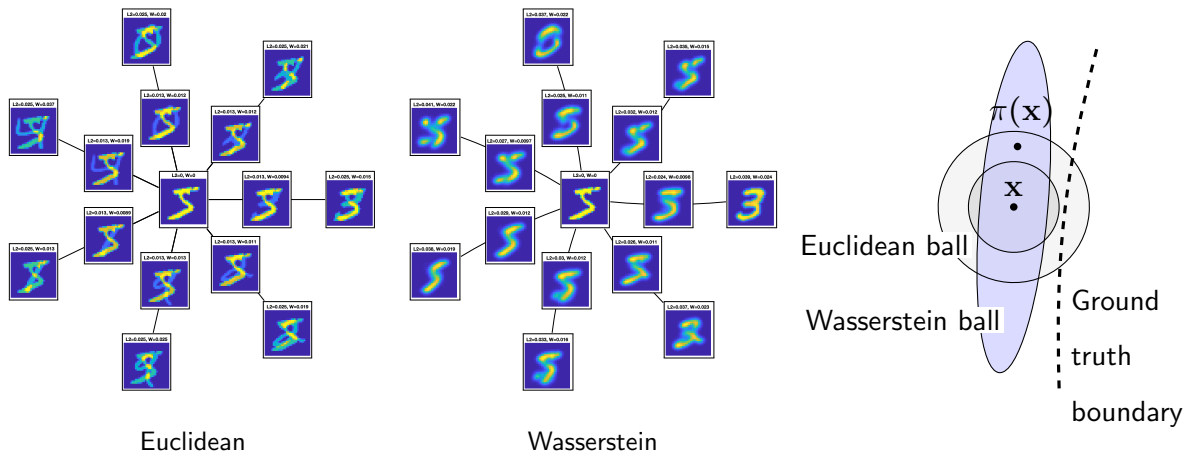


Figure 1.1: Shown are fixed-distance interpolates, measured by L^2 and Wasserstein metrics, between a source image and some other images in the MNIST dataset. On the right, we illustrate how perturbations in the isotropic neighborhood of \mathbf{x} can lead to different prediction, whereas the Wasserstein metric has the flexibility to avoid such overly general smoothness, while maintaining a semantically large neighborhood of the data.

Before presenting the details of the method, we provide a summary of the necessary background on optimal transport and the relevant notions of the Wasserstein metric below. The rest of the chapter is organized as follows, in Section 1.2 we derive the Wasserstein Ground Metric in detail and present necessary mathematical background, followed by reviewing the relevant literature of the Wasserstein metric and its uses in machine learning in Section 1.3. In Section 1.4 we present the application of the Wasserstein Ground Metric to generative modelling, and in Section 1.5 we review using the Wasserstein Ground Metric in improving deep learning models' adversarial robustness. Lastly, proofs, experimental details and additional results are relegated to Appendix 1.A.

1.2 Mathematics of optimal transport and the Wasserstein Ground Metric

In this section we review relevant mathematical properties in optimal transport theory for our framework. These properties will be used intensively throughout the chapter enabling novel application of data-driven regularization in deep learning.

Optimal transport is the mathematical field studying the minimal path, and distance of transporting an initial probability distribution ρ_0 to a final probability distribution ρ_1 with $\rho_0, \rho_1 \in \mathcal{P}_p(\mathcal{X})$. Today this is a rich field in mathematics spanning the areas of PDEs, probability theory, and analysis with applications in economics, biology and more recently machine learning [FZM15]. In the above, $\mathcal{P}_p(\mathcal{X})$ is the family of probability distribution on \mathcal{X} with bounded p th moment and the distance and path are defined according to a ground measure of distance referred to as the *ground metric*, $d_{\mathcal{X}}(\cdot, \cdot)$.

Mathematically, we define the optimal transport plan, Π and minimal distance $\mathcal{W}_{p,d_{\mathcal{X}}}(\rho_0, \rho_1)$ between probability distributions ρ_0, ρ_1 as

$$\mathcal{W}_{p,d_{\mathcal{X}}}(\rho_0, \rho_1) = \inf_{\Pi} \left\{ \left(\mathbb{E}_{(X,Y) \sim \Pi} d_{\mathcal{X}}(X, Y)^p \right)^{\frac{1}{p}} \right\}. \quad (1.1)$$

Here the infimum is taken over all joint measures $\Pi \geq 0$ with marginals

$$\int_{\Omega} \Pi(x, y) dx = \rho_0(y), \quad \int_{\Omega} \Pi(x, y) dy = \rho_1(x).$$

The definition above is due to Kantorovich and describes the \mathcal{W}_p distance where p is the exponent in the integral. We note that \mathcal{W}_p defines a proper metric named the *Wasserstein metric* between $\rho_0, \rho_1 \in \mathcal{P}_p(\mathcal{X})$.

Below we briefly review the duality structures of the Wasserstein- p metric in continuous sample space. More details are provided in [Vil09]. When $p = 1$, a particular duality structure is shown. Later we discuss that for $p = 2$, the \mathcal{W}_2 metric has a metric tensor property.

In integral form in continuous space, we have that given a sample space $\Omega \subset \mathbb{R}^d$, the Wasserstein- p metric introduces a distance between probability density functions $\rho_0, \rho_1 \in \mathcal{P}_p(\Omega)$

by

$$\mathcal{W}_p(\rho_0, \rho_1)^p = \inf_{\Pi} \int_{\Omega \times \Omega} d_{\mathcal{X}}(\mathbf{x}, \mathbf{y}) \Pi(\mathbf{x}, \mathbf{y}) d\mathbf{x} d\mathbf{y},$$

where again the infimum is taken over all joint measures $\Pi \geq 0$ with marginals

$$\int_{\Omega} \pi(\mathbf{x}, \mathbf{y}) d\mathbf{x} = \rho_0(\mathbf{y}), \quad \int_{\Omega} \pi(\mathbf{x}, \mathbf{y}) d\mathbf{y} = \rho_1(\mathbf{x}).$$

The dual problem of the Kantorovich formulation has the form

$$\begin{aligned} & \mathcal{W}_p(\rho_0, \rho_1)^p \\ &= \sup_{\Phi_0, \Phi_1 \in C(\Omega)} \left\{ \int_{\Omega} \Phi_1(\mathbf{x}) \rho_1(\mathbf{x}) - \Phi_0(\mathbf{x}) \rho_0(\mathbf{x}) d\mathbf{x} : \right. \\ & \quad \left. \Phi_1(\mathbf{y}) - \Phi_0(\mathbf{x}) \leq d_{\mathcal{X}}(\mathbf{x}, \mathbf{y}) \right\}, \end{aligned}$$

where $\Phi_0, \Phi_1: \Omega \rightarrow \mathbb{R}$ are the Lagrangian multiplier variables for the constraint of linear programming involving ρ_0, ρ_1 . Here Φ_0, Φ_1 are the so-called Kantorovich dual variables and the dual formulation is called the Kantorovich-Rubinstein duality.

1.2.1 Wasserstein-1 metric

If $p = 1$, one can show that $\Phi_1(\mathbf{x}) = \Phi_0(\mathbf{x})$. If we denote $f(\mathbf{x}) = \Phi_1(\mathbf{x})$, then the constraint condition for the duality problem has the form

$$f(\mathbf{x}) - f(\mathbf{y}) \leq d_{\mathcal{X}}(\mathbf{x}, \mathbf{y}), \quad \text{for any } \mathbf{x}, \mathbf{y} \in \Omega.$$

This gives a 1-Lipschitz condition with respect to the norm of the metric $d_{\mathcal{X}}(\mathbf{x}, \mathbf{y})$, i.e.

$$\|\nabla_{\mathbf{x}}^{d_{\mathcal{X}}} f(\mathbf{x})\| \leq 1. \tag{1.2}$$

The condition 1.2 is key for defining the WGAN framework for generative modelling (cf. [ACB17]). We explain this in depth in Section 1.4.

Yet another formulation of the Wasserstein metric via optimal control is as follows,

$$\inf_{\mathbf{m}} \left\{ \int_{\Omega} \|\mathbf{m}(\mathbf{x})\| d\mathbf{x} : \text{div}(\mathbf{m}) + \rho_1 - \rho_0 = 0 \right\}$$

where \mathbf{m} is the flux function, and div is the divergence operator depending on the ground metric $d_{\mathcal{X}}$, then the minimizer of the Wasserstein function satisfies

$$\begin{cases} \text{div}(\mathbf{m}(\mathbf{x})) = \rho_0(\mathbf{x}) - \rho_1(\mathbf{x}) \\ \frac{\mathbf{m}(\mathbf{x})}{\|\mathbf{m}(\mathbf{x})\|_{d_{\mathcal{X}}}} = \nabla_{\mathbf{x}}f(\mathbf{x}), \quad \text{when } \|\mathbf{m}(\mathbf{x})\|_{d_{\mathcal{X}}} > 0. \end{cases}$$

As we can see, the second formula in the above system satisfies the Lipschitz-1 condition, i.e. the Eikonal equation

$$\|\nabla_{\mathbf{x}}f(\mathbf{x})\| = \left\| \frac{\mathbf{m}(\mathbf{x})}{\|\mathbf{m}(\mathbf{x})\|_{d_{\mathcal{X}}}} \right\| = 1.$$

Following the direction of the flux function $\mathbf{m}(\mathbf{x})$ by the direction of $\nabla_{\mathbf{x}}f(\mathbf{x})$, one transports ρ_0 to ρ_1 . The transport direction follows the characteristic of Eikonal equation, i.e. the geodesic curve in (Ω, d) .

Next we explain the properties of the Wasserstein metric for $p = 2$ in continuous space.

1.2.2 Wasserstein-2 metric

If $p = 2$, one can relate the duality formula of Φ_1, Φ_0 with the solution of Hamilton-Jacobi equation by the Hopf-Lax formula [Vil09]. In other words, $\Phi_0(\mathbf{x}), \Phi_1(\mathbf{x})$ are the solution of Hamilton-Jacobi equation at times $t = 0, t = 1$:

$$\partial_t \Phi(t, \mathbf{x}) + \frac{1}{2} \|\nabla_{\mathbf{x}}^{d_{\mathcal{X}}} \Phi(t, \mathbf{x})\|^2 = 0.$$

The minimizer of optimal transport equation follows the form

$$\begin{cases} \partial_t \rho(t, \mathbf{x}) + \text{div}(\rho(t, \mathbf{x}) \nabla \Phi(t, \mathbf{x})) = 0 \\ \partial_t \Phi(t, \mathbf{x}) + \frac{1}{2} \|\nabla_{\mathbf{x}}^{d_{\mathcal{X}}} \Phi(t, \mathbf{x})\|^2 = 0 \end{cases} \quad (1.3)$$

with the time zero and one density solution $\rho(0, \mathbf{x}) = \rho_0(\mathbf{x}), \rho(1, \mathbf{x}) = \rho_1(\mathbf{x})$. We notice the fact that the characteristic of continuity equation and Hamilton-Jacobi equation is again the geodesics in the space Ω .

1.2.3 Wasserstein metric on graphs

We note that for the input space of raster images, the probability distribution describing each datum is defined over the finite-dimensional space of the pixel grid. We now define the Wasserstein metric over a graph: Consider the discrete pixel space $I = \{1, \dots, n\}$. The probability simplex on I is the set

$$\mathcal{P}(I) = \left\{ (p_1, \dots, p_n) \in \mathbb{R}^n : \sum_{i=1}^n p_i = 1, \quad p_i \geq 0 \right\}.$$

Here $p = (p_1, \dots, p_n)$ is a probability vector with coordinates p_i corresponding to the probabilities assigned to each node $i \in I$. The probability simplex $\mathcal{P}(I)$ is a manifold with boundary. We denote the interior by $\mathcal{P}_+(I)$. This consists of the strictly positive probability distributions, with $p_i > 0$ for all $i \in I$. To simplify the discussion, we will focus on the interior $\mathcal{P}_+(I)$.

We next define the Wasserstein-2 metric on $\mathcal{P}_+(I)$, We need to give the ground metric a notion on the discrete space. We do this in terms of a undirected graph with weighted edges, $\mathcal{G} = (I, E, \omega)$, where I is the vertex set, $E \subseteq \binom{I}{2}$ is the edge set, and $\omega = (\omega_{ij})_{i,j \in I} \in \mathbb{R}^{n \times n}$ is a matrix of edge weights satisfying

$$\omega_{ij} = \begin{cases} \omega_{ji} > 0, & \text{if } (i, j) \in E \\ 0, & \text{otherwise} \end{cases}.$$

The set of neighbors (adjacent vertices) of i is denoted by $N(i) = \{j \in I : (i, j) \in E\}$. Then the cost or distance between a node i and neighbor $j \in N(i)$ is $1/\omega_{ij}$. In the next subsection we will define the graph Wasserstein-2 metric more formally using the the Wasserstein-2 metric structure.

1.2.4 Riemannian calculus of \mathcal{W}^2

With this, we are ready to discuss the regularization term. For all of the applications of the Wasserstein regularisation we consider, we are interested in using a gradient penalty of the form $\|\nabla_{\mathbf{x}}^{\mathcal{W}^p} f\|$ where the ground metric \mathcal{W}^p is defined over discrete space. To compute

such gradient form, we consider the case where $p = 2$ in which the Wasserstein metric admits a *Riemannian structure*. This enables computing $\|\nabla_{\mathbf{x}}^{\mathcal{W}^p} f\|$ from the usual Euclidean L2 gradient. We first describe the Riemannian calculus of the Wasserstein metric generally in the continuous space and then dive into the deriving the Riemannian structure on graphs (discrete space).

\mathcal{W}^2 metric tensor for continuous space Consider again $\Omega \subset \mathbb{R}^d$ to be a compact region with the set of smooth and strictly positive densities:

$$\mathcal{P}_+(\Omega) = \left\{ \rho \in C^\infty(\Omega) : \rho(\mathbf{x}) > 0, \int_{\Omega} \rho(\mathbf{x}) d\mathbf{x} = 1 \right\}.$$

Denote by $\mathcal{F}(\Omega) := C^\infty(\Omega)$ the set of smooth real-valued functions on Ω . The tangent space of $\mathcal{P}_+(\Omega)$ is given by

$$T_\rho \mathcal{P}_+(\Omega) = \left\{ \sigma \in \mathcal{F}(\Omega) : \int_{\Omega} \sigma(\mathbf{x}) d\mathbf{x} = 0 \right\},$$

of feasible directions in the space $\mathcal{P}_+(\Omega)$. Given $\Phi \in \mathcal{F}(\Omega)$ and $\rho \in \mathcal{P}_+(\Omega)$, define

$$V_\Phi(\mathbf{x}) := -\nabla \cdot (\rho(\mathbf{x}) \nabla \Phi(\mathbf{x})) \in T_\rho \mathcal{P}_+(\Omega). \quad (1.4)$$

Here the elliptic operator (1.2.4) identifies the function Φ on Ω modulo additive constants with the tangent vector V_Φ in $\mathcal{P}_+(\Omega)$:

$$\mathcal{F}(\Omega)/\mathbb{R} \rightarrow T_\rho \mathcal{P}_+(\Omega), \quad \Phi \mapsto V_\Phi.$$

This can be seen as a proper mapping to $T_\rho \mathcal{P}_+(\Omega)$ via the the Hamilton-Jacobi formulation (1.2.2). Denote $T_\rho^* \mathcal{P}_+(\Omega) = \mathcal{F}(\Omega)/\mathbb{R}$ as the smooth cotangent space of $\mathcal{P}_+(\Omega)$. Then the L^2 -Wasserstein metric tensor on density space is defined as follows:

Definition 1.2.1 (Wasserstein-2 metric tensor). *Define the inner product on the tangent space of positive densities $g_\rho: T_\rho \mathcal{P}_+(\Omega) \times T_\rho \mathcal{P}_+(\Omega) \rightarrow \mathbb{R}$ by*

$$g_\rho^W(\sigma_1, \sigma_2) = \int_{\Omega} \nabla \Phi_1(\mathbf{x}) \cdot \nabla \Phi_2(\mathbf{x}) \rho(\mathbf{x}) d\mathbf{x},$$

where $\sigma_1 = V_{\Phi_1}$, $\sigma_2 = V_{\Phi_2}$ with $\Phi_1(\mathbf{x}), \Phi_2(\mathbf{x}) \in \mathcal{F}(\Omega)/\mathbb{R}$.

In [Laf88], $(\mathcal{P}_+(\Omega), g_\rho)$ defined in Definition 1.2.1 is named the density manifold. Following the Riemannian calculus, the gradient operator with respect to the Wasserstein-2 metric [Ott01] has the following form.

Proposition 1 (Wasserstein-2 gradient).

$$\nabla \mathcal{F}(\rho)(\mathbf{x}) = -\nabla \cdot \left(\rho \nabla \frac{\delta}{\delta \rho(\mathbf{x})} \mathcal{F}(\rho) \right),$$

and

$$\|\nabla^{\mathcal{W}^2} \mathcal{F}(\rho)\|^2 = \int \left\| \nabla \frac{\delta}{\delta \rho(\mathbf{x})} \mathcal{F}(\rho) \right\|^2 \rho(\mathbf{x}) d\mathbf{x}.$$

We note that the \mathcal{W}^2 metric can be defined using the metric tensor as,

$$\mathcal{W}^2(\rho_0, \rho_1) := \inf_{\mathbf{v}} \left\{ \int_0^1 g_p(\mathbf{v}, \mathbf{v}) dt : \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0, \rho(0) = \rho_0, \rho(1) = \rho_1 \right\}. \quad (1.5)$$

where \mathbf{v} can be seen as ∇u for a potential u . We next present the Wasserstein-2 gradient operator defined in a discrete sample space.

1.2.5 Wasserstein-2 gradient on discrete sample space

We recall the definition of discrete probability simplex with Wasserstein-2 Riemannian metric.

The normalized volume form on node $i \in I$ is given by $d_i = \frac{\sum_{j \in N(i)} \omega_{ij}}{\sum_{i=1}^n \sum_{i' \in N(i)} \omega_{ii'}}$.

The graph structure $\mathcal{G} = (I, E, \omega)$ induces a graph Laplacian matrix function.

Definition 1.2.2 (Weighted Laplacian matrix). *Given an undirected weighted graph $\mathcal{G} = (I, E, \omega)$, with $I = \{1, \dots, n\}$, the matrix function $L(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times n}$ is defined by*

$$L(p) = D^\top \Lambda(p) D, \quad p = (p_i)_{i=1}^n \in \mathbb{R}^n,$$

where

- $D \in \mathbb{R}^{|E| \times n}$ is the discrete gradient operator defined by

$$D_{(i,j) \in E, k \in V} = \begin{cases} \sqrt{\omega_{ij}}, & \text{if } i = k, i > j \\ -\sqrt{\omega_{ij}}, & \text{if } j = k, i > j \\ 0, & \text{otherwise} \end{cases}$$

- $-D^\top \in \mathbb{R}^{n \times |E|}$ is the oriented incidence matrix, and
- $\Lambda(p) \in \mathbb{R}^{|E| \times |E|}$ is a weight matrix depending on p ,

$$\Lambda(p)_{(i,j) \in E, (k,l) \in E} = \begin{cases} \frac{1}{2} \left(\frac{1}{d_i} p_i + \frac{1}{d_j} p_j \right), & \text{if } (i,j) = (k,l) \in E \\ 0, & \text{otherwise} \end{cases}.$$

The Laplacian matrix function $L(p)$ is the discrete analog of the elliptic weighted Laplacian operator $-\nabla \cdot (\rho \nabla)$ from Definition 1.2.4.

With the definition of the Laplacian we are now ready to present the discrete Wasserstein-2 metric tensor. Analogously to the continuous case, consider the tangent space of $\mathcal{P}_+(I)$ at p ,

$$T_p \mathcal{P}_+(I) = \left\{ (\sigma_i)_{i=1}^n \in \mathbb{R}^n : \sum_{i=1}^n \sigma_i = 0 \right\}.$$

Denote the space of *potential functions* on I by $\mathcal{F}(I) = \mathbb{R}^n$, and consider the quotient space

$$\mathcal{F}(I)/\mathbb{R} = \{[\Phi] \mid (\Phi_i)_{i=1}^n \in \mathbb{R}^n\},$$

where $[\Phi] = \{(\Phi_1 + c, \dots, \Phi_n + c) : c \in \mathbb{R}\}$ are functions defined up to addition of constants.

We introduce an identification map via the weighted Laplacian matrix $L(p)$ by

$$\mathbf{V}: \mathcal{F}(I)/\mathbb{R} \rightarrow T_p \mathcal{P}_+(I), \quad \mathbf{V}_\Phi = L(p)\Phi.$$

We know that $L(p)$ has only one simple zero eigenvalue with eigenvector $c(1, 1, \dots, 1)$, for any $c \in \mathbb{R}$. This is true since for $(\Phi_i)_{i=1}^n \in \mathbb{R}^n$,

$$\begin{aligned} \Phi^\top L(p)\Phi &= (D\Phi)^\top \Lambda(p)(D\Phi) \\ &= \sum_{(i,j) \in E} \omega_{ij} (\Phi_i - \Phi_j)^2 \left(\frac{1}{2} \left(\frac{1}{d_i} p_i + \frac{1}{d_j} p_j \right) \right) = 0 \end{aligned}$$

implies $\Phi_i = \Phi_j$, $(i,j) \in E$. If the graph is connected, as we assume, then $(\Phi_i)_{i=1}^n$ is a constant vector. Thus $V_\Phi: \mathcal{F}(I)/\mathbb{R} \rightarrow T_p \mathcal{P}_+(I)$ is a well defined map, linear, and one to one. By that we mean that $\mathcal{F}(I)/\mathbb{R} \cong T_p^* \mathcal{P}_+(I)$, where $T_p^* \mathcal{P}_+(I)$ is the cotangent space of $\mathcal{P}_+(I)$. This identification induces the following inner product on $T_p \mathcal{P}_+(I)$.

Definition 1.2.3 (Wasserstein-2 metric tensor). *The inner product $g_p : T_p\mathcal{P}_+(I) \times T_p\mathcal{P}_+(I) \rightarrow \mathbb{R}$ takes any two tangent vectors $\sigma_1 = \mathbf{V}_{\Phi_1}$ and $\sigma_2 = \mathbf{V}_{\Phi_2} \in T_p\mathcal{P}_+(I)$ to*

$$g_p(\sigma_1, \sigma_2) = \sigma_1^\top \Phi_2 = \sigma_2^\top \Phi_1 = \Phi_1^\top L(p) \Phi_2. \quad (1.6)$$

In other words,

$$g_p(\sigma_1, \sigma_2) := \sigma_1^\top L(p)^\dagger \sigma_2, \quad \text{for any } \sigma_1, \sigma_2 \in T_p\mathcal{P}_+(I),$$

where $L(p)^\dagger$ is the pseudo inverse of $L(p)$.

Following the inner product (1.6), the Wasserstein-2 metric on images $W : \mathcal{P}_+(I) \times \mathcal{P}_+(I) \rightarrow \mathbb{R}$ is defined using the Laplacian $L(\rho)$ by

$$\mathcal{W}^2(\rho_0, \rho_1) := \inf_{\mathbf{v}(t), \rho(t)} \left\{ \int_0^1 \mathbf{v}(t)^\top L(\rho(t)) \mathbf{v}(t) dt : \frac{\partial \rho}{\partial t} + L(\rho(t)) \mathbf{v}(t) = 0, \rho(0) = \rho_0, \rho(1) = \rho_1 \right\}. \quad (1.7)$$

The Wasserstein-2 metric on the graph introduces the following gradient operator.

Theorem 1.2.1 (Wasserstein gradient on graphs). *Given $\mathcal{F} \in C^1(\mathcal{P}_+(I))$, the gradient operator in Riemannian manifold $(\mathcal{P}_+(I), g)$ satisfies*

$$\text{grad } \mathcal{F}(p) = L(p) d_\rho \mathcal{F}(p),$$

where d is the Euclidean gradient operator.

The Laplacian matrix associated with the weighted graph \mathcal{G} is defined, depending on the input \mathbf{x} , as

$$L(\mathbf{x})_{ij} = \begin{cases} \frac{1}{2} \sum_{k \in N(i)} \omega_{ik} \left(\frac{\mathbf{x}_i}{d_i} + \frac{\mathbf{x}_k}{d_k} \right), & \text{if } i = j \\ -\frac{1}{2} \omega_{ij} \left(\frac{\mathbf{x}_i}{d_i} + \frac{\mathbf{x}_j}{d_j} \right), & \text{if } j \in N(i) \\ 0, & \text{otherwise.} \end{cases}$$

The Wasserstein metric tensor is the matrix function given by the (pseudo) inverse of the weighted Laplacian operator,

$$G_{\mathcal{W}}(x) = L(x)^{-1} \in \mathbb{R}^{n \times n}.$$

Written explicitly, the Wasserstein gradient norm squared is

$$\begin{aligned}
\|\nabla_{\mathbf{x}}^{\mathcal{W}^2} f(\mathbf{x})\|^2 &= \nabla_{\mathbf{x}} f(\mathbf{x})^\top G_{\mathcal{W}}(\mathbf{x})^{-1} \nabla_{\mathbf{x}} f(\mathbf{x}) \\
&= \nabla_{\mathbf{x}} f(\mathbf{x})^\top L(\mathbf{x}) \nabla_{\mathbf{x}} f(\mathbf{x}) \\
&= \sum_{(i,j) \in E} \omega_{ij} \left(\frac{\partial}{\partial \mathbf{x}_i} f(\mathbf{x}) - \frac{\partial}{\partial \mathbf{x}_j} f(\mathbf{x}) \right)^2 \frac{\mathbf{x}_i/d_i + \mathbf{x}_j/d_j}{2}. \tag{1.8}
\end{aligned}$$

We now describe the efficient computation of the gradient penalty term (1.8) using convolutions.

1.2.6 Efficient implementation of the Wasserstein gradient norm

To compute (1.8) in practice, we define a suitable similarity graph $\mathcal{G} = (V, E, \omega)$ for the space of images, displaying translation invariance and symmetries. First, there is the invariance with respect to pixel translations. Symmetries arise since the distance from a pixel to two spatially opposite pixels is equal. In addition, each term in the sum in (1.8) can be decomposed into the entrywise product of linear relations between values in nodes i and j . Each linear relation (e.g. $\nabla_{\mathbf{x}_i} f - \nabla_{\mathbf{x}_j} f$) is computed via a convolution to define each term over all spatial locations (pixels). Due to the symmetries of the graph a fixed kernel convolution may replace a linear product owing it to the described symmetries and invariances. For each relative neighbor direction we define a convolutional filter with the number of filters equal to the number of possible neighbor types. For the truncated similarity graph for the Wasserstein distance, the edge set E is sparse and the number of convolutional filters is reduced considerably from n^2 . We therefore can calculate all pairs $\nabla_{\mathbf{x}_i} f - \nabla_{\mathbf{x}_j} f$ with a given relative neighbor relation by passing \mathbf{x} and $\nabla_{\mathbf{x}} f$ through of kernels $K_{\mathcal{O}_1} \dots K_{\mathcal{O}_d}$. In this case a neighbor relation, is defined as the geometrical pattern between two pixels. A pixel located in position $(10, 10)$ satisfies a neighbor relation $(1, 2)$ with a pixel in location $(11, 12)$ and the same neighbor relation is satisfied between pixels $(16, 18)$ and $(17, 20)$. The neighbor relation is indeed invariant to the global position of the pixel which allows for the use of convolutions. Given a ground metric, we enumerate all non-zero neighbor relations as $\mathcal{O}_1, \dots \mathcal{O}_d$, for truncated distances the number

of relations d is much smaller than the complete edge graph. For each neighbor relation \mathcal{O}_k we associate a zero-valued kernel that equals 1 and -1 in the corresponding \mathcal{O}_k pixels, we denote the gradient kernels as $K_{\mathcal{O}_k}$. Likewise, we apply the same \mathcal{O}_k pattern, now with values $\frac{1}{2}, \frac{1}{2}$ in the corresponding neighbor pattern locations to obtain the terms $\frac{\mathbf{x}_i/d_i + \mathbf{x}_j/d_j}{2}$. For each i, j we denote the input kernels as $M_{\mathcal{O}_k}$. Applying entry-wise multiplication (\odot) and a summation collapsing all pixel locations and channels then yields an efficient and general method of calculating the Wasserstein gradient $\|\nabla_{\mathbf{x}}^{\mathcal{W}^2} f\|$ for general local topology ground metrics using convolutions, which are highly optimized in modern deep learning frameworks.

Algorithm 1 Wasserstein gradient norm $\|\nabla_{\mathbf{x}}^{\mathcal{W}^2} f\|$.

Require: The pixel graph $\mathcal{G} = (V, E, \omega)$, local weights (w_{ij}) ; neighbor relation tuples arranged symmetrically $\mathcal{O}_1 \dots \mathcal{O}_d$

Require: Euclidean gradient $\nabla_{\mathbf{x}} f$

- 1: *Wasserstein-grad* $\leftarrow 0$
 - 2: **for** neighbor relations $k = 1, \dots, d$ **do**
 - 3: Build kernel $K_{\mathcal{O}_k}$ to compute $\nabla_{\mathbf{x}_i} f - \nabla_{\mathbf{x}_{\mathcal{O}_k(i)}} f$
 - 4: Build corresponding kernel $M_{\mathcal{O}_k}$ to compute $\frac{\mathbf{x}_i}{2d_i} + \frac{\mathbf{x}_{\mathcal{O}_k(i)}}{2d_{\mathcal{O}_k(i)}}$
 - 5: $H \leftarrow K_{\mathcal{O}_k}(\nabla_X f)$
 - 6: $V \leftarrow M_{\mathcal{O}_k}(X)$
 - 7: $H \leftarrow H \odot H$ (entry-wise multiplication)
 - 8: $W \leftarrow H \odot V$
 - 9: *Wasserstein-grad* \leftarrow *Wasserstein-grad* + $\text{sum}(W)$
 - 10: **end for**
 - 11: **Return** $\|\nabla_{\mathbf{x}}^{\mathcal{W}^2} f\|^2 =$ *Wasserstein-grad*
-

1.3 Related works

In this section we review related works to the Wasserstein Ground Metric.

Wasserstein metric and gradient flows To derive the Wasserstein Ground Metric, we utilize the Riemannian structure of the metric that gives the flow formulation of the Wasserstein distance. The Wasserstein gradient flow is the result of line of works including [Ott01] that illustrates the gradient flow mathematically and physically, [Mie11] generalizes the Wasserstein gradient flow to reaction-diffusion systems and the work of [AGS05] which gives a comprehensive discussion on flows in probability spaces.

Learning and gradient flows on graphs In the discrete settings of graphs, the works of [CHL12, Maa11, SLF16, CLZ18, Li18] derive and analyze gradient flows on graphs and the corresponding metric tensors. The work of [CHL12] derives a discrete Wasserstein flow on graphs which is the base of Wasserstein Ground Metric on images. We follow the exposition of [CHL12, Li18] in Section 1.2. More generally, defining distances between graph distributions, the work of [GHL15] defines a distance on the graph based on the L1 distance of the spectral signature. The work of [ZBC11] aligns with our work and extends sparse coding to image representation using a graph prior on the images. In the discrete settings, the weighted Laplacian term in (1.2.2) can be compared with the Laplacian term in [BF12] that applies diffusion on graphs for pixel-level segmentation. In geometric deep learning one considers mappings where the input space has a rich geometric structure [BBL17]. An example is the case where the input space consists of functions defined on a graph (e.g., raster images, where the graphs are grids). One can then define convolutions based on the group structures of these graphs.

Optimal transport in machine learning Optimal transport methods including the Wasserstein metric are highly beneficial for many applications in machine learning. In the work of [SHD18] the authors provide a representation learning framework for entity relation based on the Wasserstein metric. Similar to our application, [RTG00] utilizes the Wasserstein metric as a perceptual metric for images. The Wasserstein metric is used for ensemble learning in the method of [DMM19]. In the field of inverse problems, the work of [EY18] uses optimal

transport for the seismic sensing, and [PHO18] considers the optimal transport metric for de-noising. In generative models, the influential work of [ACB17] uses the Wasserstein metric as a loss function objective, we elaborate on this in Section 1.4, on the same note, the works of [BJG17, DZS18, FZM15] apply the Wasserstein metric as a learning objective and minimize the discrepancy between distributions. Used differently, the Wasserstein metric has also been used to modify the optimization algorithm of parameterized statistical models as done in [LM18b, LLO18, MMC16, GPC18, LM18a].

1.4 Wasserstein of Wasserstein loss for learning generative models

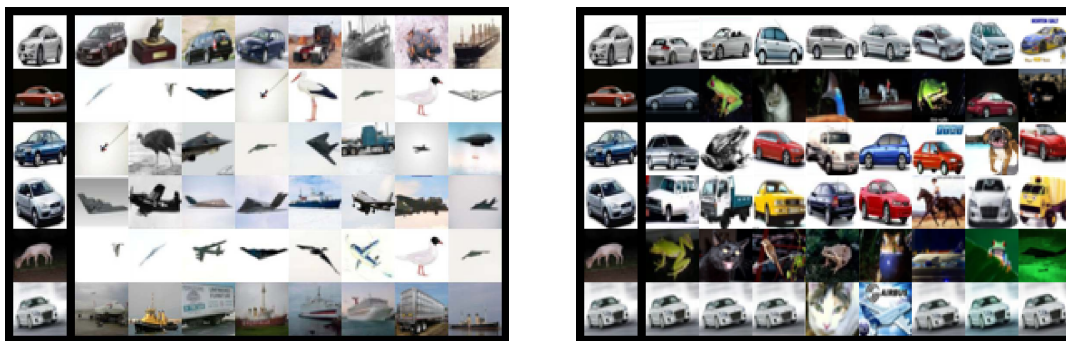
In addition to using tools from optimal transport for regularization of machine learning, in recent years optimal transport has become increasingly important in the formulation of training objectives for deep learning applications [FZM15, MMC16, ACB17]. Especially when the model output can be viewed as a probability distribution. In contrast to traditional information divergences (arising in maximum likelihood estimation), the Wasserstein distance between probability distributions incorporates the distance between samples via the ground metric of choice. In this way, it provides a continuous loss function for learning probability models supported on possibly disjoint, lower dimensional subsets of the sample space. These properties are especially useful for training implicit generative models, with a prominent example being Generative Adversarial Networks (GANs). The application of the Wasserstein metric to define the objective function of GANs is known as Wasserstein GANs (WGANs) [FZM15, ACB17, DZS18].

When training WGANs, one problem that remains is that of choosing a suitable ground metric for the sample space. The choice of the ground metric plays a crucial role in the training quality of WGANs. Usually the distance between two sample images is taken to be the mean square difference over the features, i.e., the L^2 (Euclidean) norm. This, however, does not incorporate additional knowledge that we have about the space of natural images. In order to improve training and direct focus to selected features, other Sobolev norms

in image space have been studied [AL18]. Recent works are also investigating distances based on higher level representations of the samples, which can be obtained by means of techniques such as vector embeddings [MSG17], auto-encoders, or other unsupervised and semi-supervised feature learning techniques [NJT06]. Meanwhile, as described in Section 1.1, another distance that has been very successful in comparing images, has remained unnoticed in the context of WGANs, namely the Wasserstein distance on images. In particular, the Wasserstein distance has been successful in image retrieval problems [RTG00, ZML07] and it is known to correlate well with human perception for natural images, e.g., being robust to translations and rotations [EY18, PHO18] as discussed in the introduction. See Figure 1.2 for an illustration of the Wasserstein metric. Another benefit of the Wasserstein metric on images is that it is very natural and does not require computing higher level representations of the images or any feature selection.

In this section, we propose to apply the Wasserstein metric as the objective for generative modelling with the Wasserstein Ground Metric presented in Section 1.1 as the ground metric. The Wasserstein objective defines a distance over the sample space of images and the Wasserstein Ground Metric defines a distance over the discrete space of pixels. Using the Wasserstein metric as the objective with the Wasserstein Ground Metric as the ground metric, we call our framework the Wasserstein of Wasserstein loss. At first sight, it may appear overly complicated to define a loss function of this form since computing the Wasserstein distance is already quite involved, a Wasserstein loss based on another Wasserstein Ground Metric may seem infeasible. Nonetheless, we will show that it is possible to derive an equivalent expression in the settings of gradient penalty of WGANs [PFL17]. Recall from Subsection 1.2.4, the Wasserstein-2 ground metric exhibits a metric tensor structure [Ott01, Vil09]. This enables us to introduce a Lipschitz condition based on the Wasserstein norm, rather than the L^2 norm in the gradient-penalty WGAN setting.

In this section we focus on generative models for images and specifically the WGAN formulation, but the proposed Wasserstein of Wasserstein loss function can be applied to



(a) L^2 (Euclidean) ground metric

(b) Wasserstein-2 ground metric

Figure 1.2: Source image and 9 nearest neighbors from the CIFAR-10 dataset, with respect to the L^2 (left) and Wasserstein-2 (right) ground metrics. We note that the Wasserstein-2 distance is robust to translations and rotations, and gives neighbors that are perceptually similar. In contrast, the Euclidean distance is highly sensitive and oftentimes the nearest neighbors are predominantly white images.

learning with other types of models or other types of data for which a natural distance between features can be introduced.

The rest of the section is organized as follows. In Subsection 1.4.1, we introduce the Wasserstein loss function equipped with the Wasserstein Ground Metric. Based on duality and the metric tensor of the proposed problem, we derive an equivalent practical formulation. Related works are reviewed in Subsection 1.4.2. In Subsection 1.4.3 we discuss our application to Wasserstein of Wasserstein GANs (WWGANs). Numerical experiments illustrating the benefits of the new gradient norm penalty are provided in Subsection 1.4.4. Lastly, we provide a discussion of the Wasserstein of Wasserstein loss in Subsection 1.4.5.

1.4.1 Wasserstein of Wasserstein loss

In this section, we introduce the Wasserstein Ground Metric for the Wasserstein loss function. A motivating example is presented to demonstrate the utility of the proposed model.

1.4.1.1 Wasserstein loss

Consider a metric sample space $(\mathcal{X}, d_{\mathcal{X}})$. Recall from Section 1.2 that the Wasserstein- p distance can be defined as follows. Given a pair $\rho_0, \rho_1 \in \mathcal{P}_p(\mathcal{X})$ of probability densities with finite p -th moment, let

$$\mathcal{W}_{p,d_{\mathcal{X}}}(\rho_0, \rho_1) = \inf_{\Pi} \left\{ \left(\mathbb{E}_{(X,Y) \sim \Pi} d_{\mathcal{X}}(X, Y)^p \right)^{\frac{1}{p}} \right\}, \quad (1.9)$$

where Π is a joint distribution of (X, Y) with marginals $X \sim \rho_0, Y \sim \rho_1$. We note that \mathcal{W}_p crucially depends on the choice of a distance function $d_{\mathcal{X}}: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ (ground metric) on sample space.

In practice, the sample space \mathcal{X} is typically very high dimensional, sometimes even being an (infinite dimensional) Banach space. We focus on the case where \mathcal{X} is the space of images, which can be regarded as a density space over pixels, i.e., $\mathcal{X} = \mathcal{P}(\Omega)$, where $\Omega = [0, M] \times [0, M]$ is a discrete grid of pixels. With this in mind, we will define the distance function between pixels $d_{\Omega}: \Omega \times \Omega \rightarrow \mathbb{R}_+$ according to Subsection 1.2.5.

1.4.1.2 Wasserstein loss function with Wasserstein Ground Metric

We now introduce the Wasserstein of Wasserstein loss. Here, the first ‘Wasserstein’ refers to the Wasserstein loss function over probability distributions on the space of images. The second ‘Wasserstein’ refers to the ground metric of this loss function. It is chosen as the Wasserstein distance over the space of images defined as histograms over pixels, having a ground metric over pixel locations.

That is, a raster image can be viewed as a 2D histogram with each pixel representing a bin for each channel. By defining a ground metric between pixels (e.g., the physical distance between pixels), we introduce the Wasserstein distance between images. This serves as the new ground metric for defining a Wasserstein distance between probability distributions over images. See Figure 1.3.

As mentioned in the introduction, the Wasserstein distance is also known as the Earth

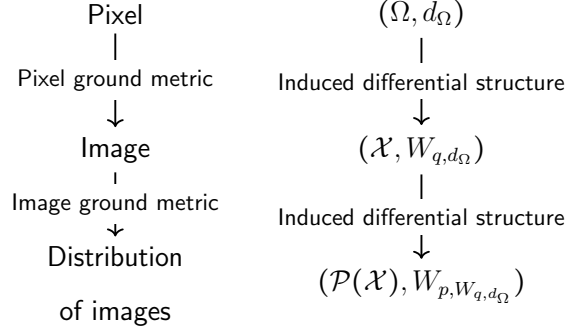


Figure 1.3: Illustration of Wasserstein- p loss function with Wasserstein- q ground metric.

Mover’s distance and is known as an effective metric in distinguishing images [RTG00]. Motivated by this fact, we use the Earth Mover’s distance (of images) as the ground metric,

$$\begin{aligned}
 d_{\mathcal{X}}(X, Y) &:= W_{q, d_{\Omega}}(X, Y) \\
 &= \inf_{\pi} \left\{ \left(\mathbb{E}_{(x, y) \sim \pi} d_{\Omega}(x, y)^q \right)^{\frac{1}{q}} \right\},
 \end{aligned} \tag{1.10}$$

where π is a joint distribution of (x, y) with marginals $x \sim X, y \sim Y$ both being images viewed as histograms over pixels. Here $d_{\mathcal{X}} = W_{q, d_{\Omega}}(x, y)$ is named Wasserstein- q ground metric. It is defined with the pixel ground metric $d_{\Omega}: \Omega \times \Omega \rightarrow \mathbb{R}_+$ assigning distances to pairs of pixels.

In this section, combining the above approaches, we obtain a Wasserstein- p distance with Wasserstein- q ground metric as the loss function for training.

Definition 1.4.1. *Given a probability model $\{\mathbb{P}_G: G \in \Theta\} \subseteq \mathcal{P}_p(\mathcal{X})$ and a data distribution $\mathbb{P}_r \in \mathcal{P}_p(\mathcal{X})$, we propose the minimization problem*

$$\inf_G W_{p, W_{q, d_{\Omega}}}(\mathbb{P}_G, \mathbb{P}_r), \tag{1.11}$$

where $\mathcal{P}_p(\mathcal{X})$ is the set of densities with finite p -th moment, $W_{p, d_{\mathcal{X}}}$ is defined by (1.1) and $W_{q, d_{\Omega}}$ is given by (1.10).

The next example illustrates the difference between the proposed Wasserstein of Wasserstein loss and the Wasserstein loss with L^2 ground metric.

Motivating example. Consider the distribution $\mathbb{P}_r = \delta_X$ which assigns probability one to a single image X . Suppose the generative model attempts to estimate this via a distribution

of the form $\mathbb{P}_G = \delta_Y$ which assigns probability one to a fake image Y . Now suppose that $X = \delta_x$, $Y = \delta_y$ are images with intensity 1 on pixel locations x , y , respectively, and intensity zero elsewhere. See Figure 1.4. In this case we have

$$W_{p,d_{\mathcal{X}}}(\mathbb{P}_r, \mathbb{P}_G) = d_{\mathcal{X}}(X, Y).$$

We check the following choices of the ground metric $d_{\mathcal{X}}$ between images X and Y .

1. Wasserstein-2 ground metric:

$$d_{\mathcal{X}}(X, Y) = W_{2,d_{\Omega}}(X, Y) = d_{\Omega}(x, y);$$

2. L^2 (Euclidean) ground metric:

$$d_{\mathcal{X}}(X, Y) = d_{L^2}(X, Y) = \begin{cases} 0 & \text{if } x = y \\ \text{constant} & \text{if } x \neq y \end{cases}.$$

We see that the Wasserstein distance with L^2 ground metric will assign two distant pixels the same cost as two adjacent pixels. This results in a highly discontinuous distance that is sensitive to single pixel translations! To make matters worse, in the case of continuous domain images, the L^2 distance will be infinite for all non-overlapping pixels. On the other hand, the Wasserstein or Wasserstein loss function is continuous with respect to continuous change of pixels in images. For learning image models with low dimensional support, the Wasserstein or Wasserstein loss function is still well defined, while the Wasserstein loss with the L^2 ground metric function is ill-posed.

1.4.1.3 Duality formulation and properties

The computation required for the Wasserstein or Wasserstein loss function as stated in the previous section is unfeasible. To compute (1.11) one needs to handle a linear programming computation at both the level of probability distributions over images and individual images over pixels.

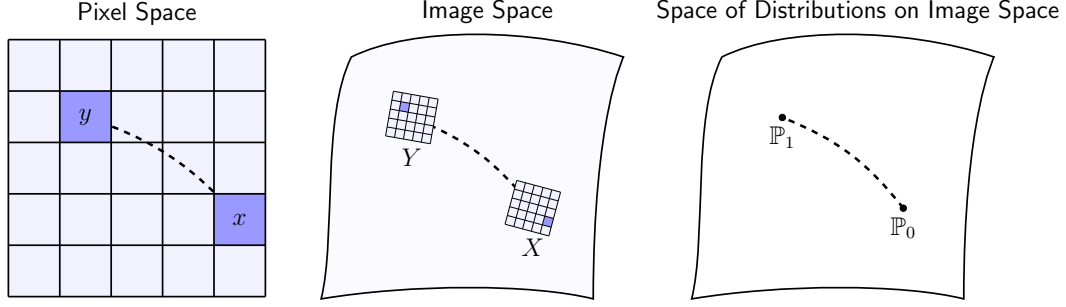


Figure 1.4: Depending on how we measure distances between pixel locations, the distance between images will be determined, and this in turn will determine how distances are measured between probability distributions.

In this section, we present the Kantorovich duality formulation of Wasserstein of Wasserstein loss function with $p = 1$ and $q = 2$. As is done for Wasserstein GANs [ACB17], we consider an equivalent Lipschitz-1 condition, which can be practically applied in the framework of GANs.

Theorem 1.4.1 (Duality of Wasserstein of Wasserstein loss function). *The Wasserstein-1 loss function over Wasserstein-2 ground metric has the following equivalent formulation:*

$$\begin{aligned}
 & W_{1,W_2,d_\Omega}(\mathbb{P}_G, \mathbb{P}_r) \\
 &= \sup_{f \in \mathcal{C}(\mathcal{X})} \left\{ \mathbb{E}_{X \sim \mathbb{P}_G} f(X) - \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_r} f(\mathbf{x}) : \right. \\
 & \quad \left. \int_{\Omega} \|\nabla_{\mathbf{z}} \delta_{\mathbf{x}} f(\mathbf{x}(\mathbf{z}))\|_{d_\Omega}^2 \mathbf{x}(\mathbf{z}) d\mathbf{z} \leq 1 \right\}, \tag{1.12}
 \end{aligned}$$

where $\nabla_{\mathbf{z}}$ is the gradient operator in pixel space Ω and $\delta_{\mathbf{x}}$ is the L^2 gradient in image space \mathcal{X} .

Proof of Theorem 1.4.1:

The result is from the duality of Wasserstein-1 metric, together with the Wasserstein-2 metric induced gradient operator. Using the Wasserstein-1 metric we can apply the Kantorovich duality reviewed in Section 1.2:

$$\mathcal{W}_{1,d_{\mathcal{X}}}(\mathbb{P}_0, \mathbb{P}_1) = \sup_f \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_0} f(\mathbf{x}) - \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_1} f(\mathbf{x}),$$

where the supremum is taken among all $f: \mathcal{X} \rightarrow \mathbb{R}$ satisfying a 1-Lipschitz condition with respect to the ground metric $d_{\mathcal{X}}$, i.e.,

$$\|\nabla_{\mathbf{x}}^{d_{\mathcal{X}}} f\| \leq 1. \quad (1.13)$$

Second, consider the ground metric given by the Wasserstein-2 metric $d_{\mathcal{X}} = \mathcal{W}_{2,d_{\Omega}}$ with ground metric d_{Ω} of pixel space. Then the gradient operator in $(\mathcal{X}, d_{\mathcal{X}})$ is the Wasserstein-2 gradient, i.e.,

$$\nabla^{\mathcal{W}^2} f(\mathbf{x}) = -\nabla_{\mathbf{z}} \cdot (\mathbf{x}(\mathbf{z}) \nabla_{\mathbf{z}} \delta_{\mathbf{x}} f(\mathbf{x})(\mathbf{z})).$$

The 1-Lipschitz condition for $(\mathcal{X}, d_{\mathcal{X}})$ in (1.13) gives $\|\nabla_{\mathbf{x}}^{\mathcal{W}_{2,d_{\Omega}}} f\| \leq 1$, i.e.,

$$(\nabla f(\mathbf{x}), \nabla f(\mathbf{x}))_{\mathcal{W}_{2,d_{\Omega}}} \leq 1.$$

It is rewritten as the following integral of the Lipschitz-1 condition w.r.t. the Wasserstein Ground Metric:

$$\int_{\Omega} \|\nabla_{\mathbf{z}} \delta_{\mathbf{x}} f(\mathbf{x})(\mathbf{z})\|_{d_{\Omega}}^2 \mathbf{x}(\mathbf{z}) d\mathbf{z} \leq 1.$$

Combining the above facts, we derive the formula for Wasserstein of Wasserstein loss function. □

The maximizer f in (1.12) corresponds to an Eikonal equation in image space $(\mathcal{X}, \mathcal{W}_{2,d_{\Omega}})$. In other words, the Lipschitz-1 condition in Wasserstein norm has the form

$$\int_{\Omega} \|\nabla_{\mathbf{z}} \delta_{\mathbf{x}} f(\mathbf{x})(\mathbf{z})\|_{d_{\Omega}}^2 \mathbf{x}(\mathbf{z}) d\mathbf{x} = 1.$$

We call this equation the Wasserstein Eikonal equation.

Here the characteristic curve of our Eikonal equation is the geodesic curve in Wasserstein space $(\mathcal{X}, \mathcal{W}_{2,d_{\Omega}})$. The characteristic curve of geodesics in Wasserstein space is again a geodesic in pixel space (Ω, d_{Ω}) . We call this fact the *double characteristic property*. This is illustrated in Figure 1.4. In contrast, the characteristic of geodesics in L^2 space does not depend on pixel space. In the experiments section, we show that with the double characteristic property, the discriminator is more continuous with respect to translations in pixel space, and is robust with respect to spatially independent noise added to the samples.

1.4.2 Relevant literature for the Wasserstein of Wasserstein loss

In this subsection, we review additional literature related to the Wasserstein of Wasserstein objective formulation.

Ground metric for function space. Banach GAN [AL18] pointed out the importance of the ground metric in training with the Wasserstein loss. They apply Sobolev norms and their induced gradient operator. In contrast, we apply the optimal transport induced operator [Ott01, Vil09]. The gradient operator depends on the new ground metric structure on sample space. We demonstrate that the optimal transport gradient provides for a practical 1-Lipschitz condition for training Wasserstein GANs.

Wasserstein natural gradients. Recent work also investigates natural gradients based on the Riemannian structures derived from optimal transport [LM18a]. In this case, optimal transport serves to define an optimization method, rather than a loss function. This approach has also been applied to the training of GANs, where it leads to an iterative regularizer for the generator [LLO18].

1.4.3 Wasserstein of Wasserstein GANs

In this subsection we apply the Wasserstein of Wasserstein loss function to implicit generative models.

1.4.3.1 Background

We start by reviewing generative adversarial networks (GAN). GANs are a deep learning approach to generative modelling that has demonstrated significant potential in the realm of image and text synthesis [YZW17, MCY18]. The GAN model is composed of two competing agents: A discriminator and a generator. At each training step the generator produces synthesized images and the discriminator is given a batch of real and synthesized images

to be classified as real or fake. The generator is trained to maximize the predictions of the discriminator while the discriminator is trained to classify generated images aside from real images. At the end of training the generator has learned how to trick the discriminator and ideally also estimate the underlying data distribution.

Mathematically if we define a trainable generative model \mathbb{P}_G and discriminator D , the GAN objective formulation is as follows:

$$\min_{\mathbb{P}_G} \max_D \left\{ \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_r} \log(D(\mathbf{x})) + \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_G} \log(1 - D(\mathbf{x})) \right\}. \quad (1.14)$$

Here \mathbb{P}_r is the true, or real, data distribution. The distribution \mathbb{P}_G is defined in terms of a generator parameterized by $\theta \in \mathbb{R}^d$. Let the generator be given by $G_\theta: \mathbb{R}^m \mapsto \mathcal{X}$; $\xi \mapsto \mathbf{x} = G(\theta, \xi)$. This takes a noise sample $\xi \sim p(z) \in \mathcal{P}_2(\mathbb{R}^m)$ to an output sample with density given by $\mathbf{x} = G(\theta, \xi) \sim \rho(\theta, \mathbf{x}) = \mathbb{P}_G$. Here \mathbb{R}^d is the parameter space, \mathbb{R}^m is the latent space, and \mathcal{X} is the sample space.

The approach described above was found to suffer from difficulties at training, including lack of convergence and mode collapse, a phenomenon where the distribution \mathbb{P}_G restricts to estimate a proper subset of \mathbb{P}_r . The above-mentioned challenges are often the result of the discontinuous nature of the loss in (1.14), and were also considered by [BJG17]. To resolve such problems, [ACB17] proposed to use the Wasserstein metric with Euclidean ground metric as the objective, formulated as

$$\begin{aligned} & \min_{\mathbb{P}_G} W_{1,L^2}(\mathbb{P}_G, \mathbb{P}_r) \\ &= \min_{\mathbb{P}_G} \sup_{f \in C(\mathcal{X})} \left\{ \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_G} f(\mathbf{x}) - \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_r} f(\mathbf{x}) : \right. \\ & \quad \left. \|\nabla_{\mathbf{x}} f\|_2 \leq 1 \right\}. \end{aligned} \quad (1.15)$$

The Lipschitz condition in (1.15) was enforced via weight-clipping, ensuring $\|\nabla_{\mathbf{x}} f\|_2 < C_0$, where C_0 is a constant. While now providing GAN with a continuous loss, WGAN with weight-clipping was noted to suffer from cyclic behavior and instability which was improved by [GAA17] by changing the Lipschitz enforcing condition from hard weight-clipping to a

soft gradient penalty term,

$$\begin{aligned} \min_{\mathbb{P}_G} \sup_{f \in C(\mathcal{X})} & \left\{ \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_G} f(\mathbf{x}) - \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_r} f(\mathbf{x}) \right. \\ & \left. + \lambda \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_{\text{interp}}} (\nabla_{\mathbf{x}} f(\mathbf{x}) - 1)^2 \right\}. \end{aligned} \quad (1.16)$$

Here $\mathbb{P}_{\text{interp}}$ is a linear interpolation between \mathbb{P}_r and \mathbb{P}_G , and λ is fixed. The gradient penalty term in (1.16) is not in full compliance with the Kantorovich duality of the problem as it also penalizes a discriminator of Lipschitz constants smaller than 1. To remedy this issue, [PFL17] replace the gradient penalty term by

$$\lambda \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_{\text{interp}}} (\max(\nabla_{\mathbf{x}} f(\mathbf{x}) - 1, 0))^2.$$

We now derive our formulation that improves current methods which are based on the L^2 ground metric. Following Theorem 1.4.1, the Wasserstein of Wasserstein loss function can be rewritten to give the optimization problem

$$\begin{aligned} & \min_{\mathbb{P}_G} \mathcal{W}_{1, \mathcal{W}_2, d_\Omega}(\mathbb{P}_G, \mathbb{P}_r) \\ & = \min_{\mathbb{P}_G} \sup_{f \in C(\mathcal{X})} \left\{ \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_G} f(\mathbf{x}) - \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_r} f(\mathbf{x}) : \right. \\ & \quad \left. \|\nabla_{\mathbf{x}}^{\mathcal{W}_2, d_\Omega} f(\mathbf{x})\| \leq 1 \right\}. \end{aligned}$$

The above formulation is suitable for training GANs. Here we call the dual variable, f , the discriminator, while G is the generator. In the setting of GANs, neural networks are used to approximate the discriminator and generator, giving

$$\begin{aligned} & \min_{\theta} \sup_{\phi} \left\{ \mathbb{E}_{\xi \sim p(z)} f_{\phi}(g(\theta, \xi)) - \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_r} f_{\phi}(\mathbf{x}) : \right. \\ & \quad \left. \int_{\Omega} \|\nabla_{\mathbf{z}} \delta_{\mathbf{x}} f_{\phi}(\mathbf{x})(\mathbf{z})\|_{d_\Omega}^2 \mathbf{x}(\mathbf{z}) d\mathbf{z} \leq 1 \right\}. \end{aligned}$$

Here the generator G is expressed as a neural network with parameters $\theta \in \Theta$, and the discriminator is approximated by a neural network with parameters $\phi \in \Phi$. Our approach implements the 1-Lipschitz condition in terms of the Wasserstein gradient operator.

1.4.3.2 Discretization

We follow the formulas presented in Subsection 1.2.5 that enables computing the discrete version of the Wasserstein-2 gradient. In practice, the image space \mathcal{X} is not infinite dimensional, although in vision problems the dimension may be vast ($\mathcal{X} = \mathbb{R}^{28 \times 28}$ or $\mathbb{R}^{32 \times 32 \times 3}$ for MNIST or CIFAR-10). We point to the derivation to Subsection 1.2.5 and present the Riemannian structure of the discrete \mathcal{W}^2 again here for convenience:

Proposition 2 (Wasserstein gradient on pixel space graph). *Given a pixel space graph \mathcal{G} , the gradient of $f \in C^1(\mathcal{X})$ w.r.t. (\mathcal{X}, W) satisfies*

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = L(\mathbf{x}) \nabla_{\mathbf{x}} f(\mathbf{x}),$$

where $\nabla_{\mathbf{x}}$ is the Euclidean gradient operator, and $L(\mathbf{x}) \in \mathbb{R}^{n \times n}$ is the weighted Laplacian matrix defined as

$$L(\mathbf{x})_{ij} = \begin{cases} \frac{1}{2} \sum_{k \in N(i)} \omega_{ik} \left(\frac{\mathbf{x}_i}{d_i} + \frac{\mathbf{x}_k}{d_k} \right) & \text{if } i = j; \\ -\frac{1}{2} \omega_{ij} \left(\frac{\mathbf{x}_i}{d_i} + \frac{\mathbf{x}_j}{d_j} \right) & \text{if } j \in N(i); \\ 0 & \text{otherwise.} \end{cases}$$

Moreover, the 1-Lipschitz condition w.r.t. (\mathcal{X}, W) , $\|\nabla_{\mathbf{x}}^{\mathcal{W}^2} f(\mathbf{x})\| \leq 1$, is equivalent to

$$\nabla_{\mathbf{x}} f(\mathbf{x})^\top L(\mathbf{x}) \nabla_{\mathbf{x}} f(\mathbf{x}) \leq 1.$$

Remark 1. We observe that the 1-Lipschitz condition is exactly the discrete analog of the one in equation (1.12),

$$\|\nabla_{\mathbf{x}}^{\mathcal{W}^2} f(\mathbf{x})\| = \nabla_{\mathbf{x}} f(\mathbf{x})^\top L(\mathbf{x}) \nabla_{\mathbf{x}} f(\mathbf{x}) = \sum_{(i,j) \in E} \omega_{ij} (\nabla_{\mathbf{x}_j} f(\mathbf{x}) - \nabla_{\mathbf{x}_i} f(\mathbf{x}))^2 \frac{\mathbf{x}_i/d_i + \mathbf{x}_j/d_j}{2} \leq 1.$$

Note that the Wasserstein gradient written in this form can be compared with the graph Laplacian on images [BF12, ZBC11].

1.4.3.3 Wasserstein gradient regularization in GANs

We next adopt the gradient penalty into the loss function (cf. [PFL17, GAA17]) as

$$\min_{\theta} \sup_{\phi} \left\{ \mathbb{E}_{\xi \sim p(z)} f_{\phi}(g(\theta, \xi)) - \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_r} f_{\phi}(\mathbf{x}) + \lambda \mathbb{E}_{\hat{\mathbf{x}} \sim \hat{\mathbb{P}}} \left(\sqrt{\nabla_{\mathbf{x}} f_{\phi}(\hat{\mathbf{x}})^{\top} L(\hat{\mathbf{x}}) \nabla_{\mathbf{x}} f_{\phi}(\hat{\mathbf{x}})} - 1 \right)^2 \right\},$$

where λ is chosen as a large constant and $\hat{\mathbb{P}}$ is the distribution of $\hat{\mathbf{x}}$ taken to be the uniform on “Euclidean” lines connecting points drawn from \mathbb{P}_G and \mathbb{P}_r . Our WWGAN training method is summarized in Algorithm 2.

Remark 2. *In practice, we may want to use images of un-normalized intensity, therefore the gradient penalty needs to account for change of total intensity. As proposed by [Li18], we consider*

$$\tilde{L}(\mathbf{x}) = \alpha \mathbf{1}\mathbf{1}^T + L(\mathbf{x}). \tag{1.17}$$

Here $\mathbf{1} = (1, \dots, 1)^T \in \mathbb{R}^n$ is a constant vector. In Appendix 1.A.3, we show how this adds one direction to the original tensor. Compared to $L(\mathbf{x})$ defined in the probability simplex, $\tilde{L}(\mathbf{x})$ is defined in the positive orthant. In the algorithm, we simply replace L by \tilde{L} for un-normalized intensity.

1.4.4 Experiments

In this subsection, we present experiments demonstrating the effects and utility of WWGAN. We use the CIFAR-10 and 64×64 cropped-CelebA image datasets. In both experiments the discriminator is a convolutional neural network with 3 hidden layers and leaky ReLU activations. For the generator we utilize a network with 3 hidden de-convolution layers and batch normalization [IS15]. The dimensionality of the latent variable of the generator is set at 128. Batch normalization is not applied to the discriminator, in order to avoid dependencies when computing the gradient penalties. The model is then trained with the ADAM optimizer with fixed parameters $(\beta_1, \beta_2) = (0.9, 0)$. More implementation details are provided in Appendix 1.A.3.

Algorithm 2 WWGAN Gradient Penalty.

Require: Gradient penalty coefficient λ , discriminator iterations per generator iteration $n_{disc.}$, batch size m , ADAM hyperparameters α, β_1, β_2 , initial discriminator and generator parameters ϕ_0 and θ_0 , L matrix-function from graph structure for image space $\mathcal{G} = (V, E, \omega)$.

```
1: while  $\theta$  has not converged do
2:   for  $t = 1, \dots, n_{disc.}$  do
3:     for  $i = 1, \dots, m$  do
4:       Sample real data  $\mathbf{x} \sim \mathbb{P}_r$ , latent variable  $\xi \sim p(z)$ , a random number  $\epsilon \sim U[0, 1]$ .
5:        $\tilde{\mathbf{x}} \leftarrow G_\theta(\xi)$ 
6:        $\hat{\mathbf{x}} \leftarrow \epsilon \mathbf{x} + (1 - \epsilon) \tilde{\mathbf{x}}$ 
7:        $M^{(i)} \leftarrow D_\phi(\tilde{\mathbf{x}}) - D_\phi(\mathbf{x}) + \lambda(\sqrt{\nabla_{\hat{\mathbf{x}}} D_\phi(\hat{\mathbf{x}})^T L(\tilde{\mathbf{x}}) \nabla_{\hat{\mathbf{x}}} D_\phi(\hat{\mathbf{x}})} - 1)^2$ 
8:     end for
9:      $\phi \leftarrow \text{Adam}(\nabla_\phi \frac{1}{m} \sum_{i=1}^m M^{(i)}, \phi, \alpha, \beta_1, \beta_2)$ 
10:  end for
11:  Sample a batch of latent variables  $\{\xi^i\}_{i=1}^m \sim p(z)$ 
12:   $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^m -D_\phi(G_\theta(\xi)), \theta, \alpha, \beta_1, \beta_2)$ 
13: end while
```

Figure 1.7 shows that in terms of computation time and quality of the generated images as measured by the Fréchet Inception Distance (FID), WWGAN is comparable to state of the art WGAN-GP. Next, we take a look at the properties of the trained discriminators, which also serves to probe the shape of the probability densities over images defined by generators.

1.4.4.1 Perturbation stability

In this experiment we investigate how the discriminator trained with WWGAN on images benefits from the properties of the Wasserstein Ground Metric. Specifically, we test whether the discriminator trained with the new gradient penalty is more continuous with respect to natural variations of the images. Natural variabilities are continuous transformations of natural images that result in natural looking images, such as translations and rotations. If the transformations are applied gradually, one should expect to observe only gradual changes in the discriminator. The experiment is illustrated in Figure 1.5, where a randomly selected image from the CIFAR-10 dataset is gradually shifted vertically, shifting all pixels a single pixel downward at each step. In the figure, the sequence of shifted images is passed through the WWGAN and the WGAN-GP discriminators, which had been trained with their respective loss to reach an FID value of 40 for the generator. We observe with our WWGAN model, the discriminator values change continuously with the translation of the input image. In contrast, this type of continuity is not observed in models that are trained with the Euclidean Lipschitz condition. We note that WWGAN assigns a positive value to the image and gradually decreases to the end limit when the entire image is shifted away. Unlike WWGAN, WGAN-GP is highly sensitive to perturbations in image space and oscillates wildly, assigning highly positive (real label) and negative (fake labels) to images shifted less than 2 pixels away. We observed the same type of behavior across all images tested, as reported in Table 1.1.

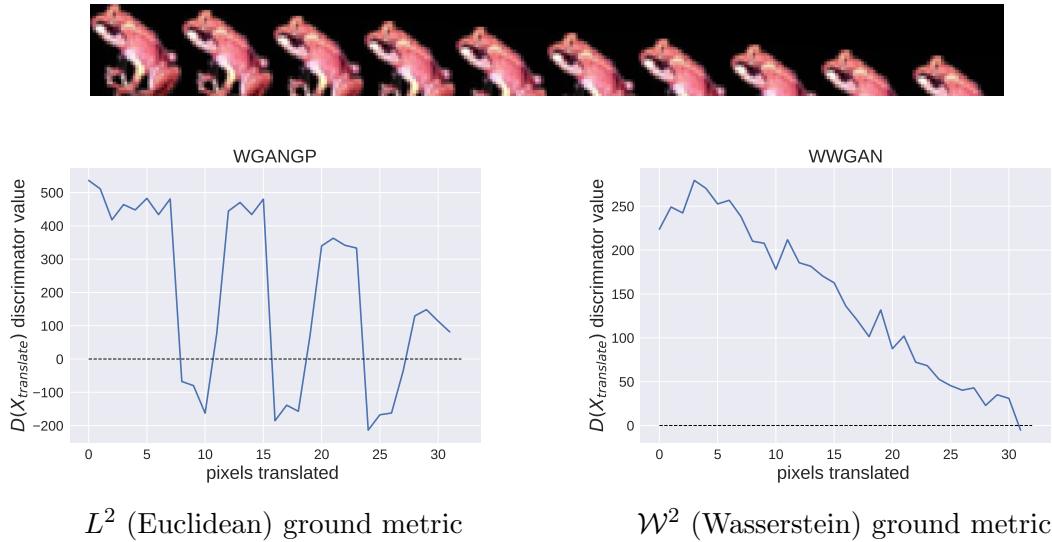


Figure 1.5: Discriminator for CIFAR-10 images translated by a vertical shift from 0 (no shift) to 32 pixels (complete image). The WWGAN discriminator is continuous to natural perturbations, e.g., vertical translation. WGAN-GP discriminator exhibits unpredictable behavior for small vertical perturbations, oscillating between real (positive values) and fake (negative values) labels. Both WWGAN, WGAN-GP discriminators tested were trained identically to reach an FID value of 40.

Method	Total variation (normalized)	zero-crossings
WGAN-GP	5.36	7.07
WWGAN	4.02	0.65

Table 1.1: For each image of the CIFAR-10 testing set we construct a vertical translation sequence and evaluate it on the discriminator of WWGAN and WGAN-GP. Normalized total variation and zero-crossing are computed for each curve and the average is reported. It is observed that WGAN-GP is more oscillatory than WWGAN.

1.4.4.2 Discriminator robustness to noise

In this experiment, we test the robustness of the discriminator to RGB salt and pepper noise, i.e., every pixel has a probability to be changed to either 0 or 1. In the plot 15% of the pixels are modified. We trained GANs with WGAN-GP and WWGAN until reaching an FID score of 40. We then measure the values of the trained discriminators on real images with RGB salt and pepper noise. In Figure 1.6, we see that WGAN-GP has separate clusters for noisy and clean images, while WWGAN is more robust to the noise and assigns more consistent values to all images.

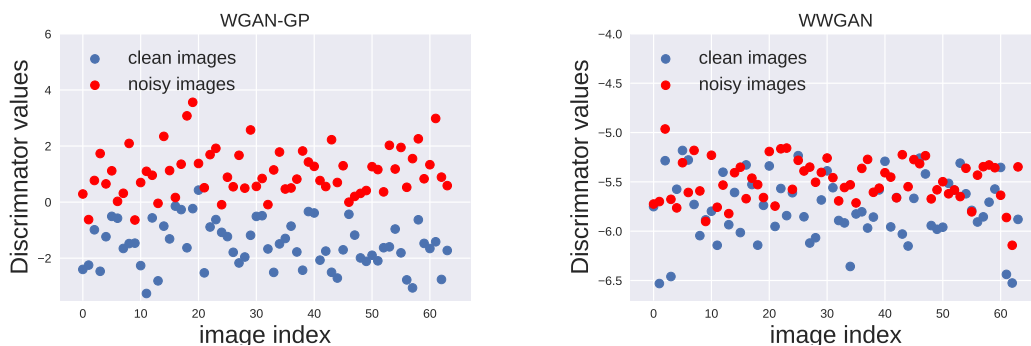


Figure 1.6: Robustness of the discriminator to noise on real CIFAR-10 images. The noise is RGB salt and pepper, where 15% of the pixels are modified. The WGAN-GP discriminator values cluster according to noise, giving different values to clean and noisy real images. The WWGAN discriminator is more robust to noise, and changes relatively little.

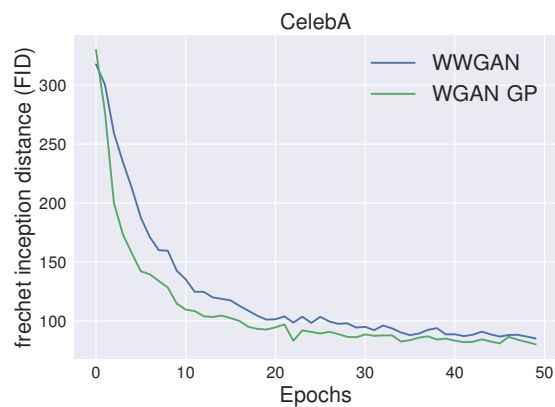


Figure 1.7: WWGAN gives comparable results with state of the art WGAN-GP training in terms of the FID of generated images. In terms of computation time, the overhead of WWGAN is negligible, with average epoch wall-clock times of 218.1 s and 236.9 s, respectively, in our experiments.

1.4.5 Discussion

In Section 1.4 we proposed a Wasserstein loss function with Wasserstein Ground Metric for learning generative models. The Wasserstein Ground Metric introduces a manifold structure into the sample space of the model and allows us to introduce meaningful priors to the learning model. Experiments demonstrate that this approach can contribute to making the generator and discriminator in GANs more stable with respect to noise and the natural variability of image data.

We consider the Wasserstein of Wasserstein loss an important advance at a conceptual level. With a clear physical intuition. Namely, it corresponds to physical displacement or translation in pixel space. This translates to continuity in image space, and changes the distribution over images accordingly. The double characteristic property of the Wasserstein Eikonal equation reflects this intuition analytically. We regard it as surprising that this high level approach can be translated to practical computational methods. Remarkably, our approach has a very small additional computational cost over the standard Wasserstein loss function with L^2 (Euclidean) ground metric.

1.5 Wasserstein Tikhonov regularization in image classification

In this section we apply the Wasserstein Ground Metric to improve *discriminative* models via a more natural notion of distance between samples.

1.5.1 Introduction

The sensitivity of trained discriminative models to small perturbations of the input data has become a reason of concern and an important topic of research in recent years [SZS14, NYC15]. In particular, it has been observed that neural networks which have been trained to have good test performance can be fooled when the inputs are slightly perturbed in a way that is imperceptible to humans. This indicates a poor generalization ability, and specifically,

that the solutions found with naive training and validation techniques are not capturing appropriate smoothness priors over the input space. A number of recent works have proposed approaches to improve robustness to perturbations [CBG17, LLD18, SKC18, WLS18, FOA19], while a complementary line of work probes the limitations of trained networks [SYZ19] and develops strategies to generate adversarial attacks [CKB17, MFF16, MFF17, SHS19].

Intuitively, a smoother function at fixed training accuracy should be more robust to perturbations of the input, including adversarial attacks. Therefore, one strategy is to train the discriminative function with smoothness regularizers, such as noise added to the training examples (adversarial or random) or penalizing the norm of the gradient with respect to the inputs. We note, however, that the notion of a “small” perturbation will strongly depend on how we decide to measure distances in the space of inputs. The gradient and its norm depend on the geometric structure that endows the input space.

While it is convenient to use the L^2 metric (Euclidean), it is well understood that many data types of interest are not Euclidean. In particular, the L^2 metric does not measure distances between images in the way that we perceive them. Changes that humans consider small, might correspond to changes that the classifier considers to be large in this metric. Moreover, it is clear that a discriminative function on image data should be more stable in certain directions and more variable in other directions. This distinction is not well captured by isotropic smoothness regularizers.

To construct more effective smoothness regularizers, two general approaches come to mind: 1) Measure distances in a metric representation of the raw inputs, $d_\phi(x, y)^2 = \sum_j |\phi(x)_j - \phi(y)_j|^2$, where ϕ is some feature representation function that might be trained separately from or together with the discriminative task. Examples in this direction include preprocessing of the inputs by downsampling [GRC18], autoencoders, and approaches that regularize intermediate representations within the neural network that is being trained for the discriminative task, such as injecting noise in the layers of a ResNet [WYS18]. 2) Measure distances directly on the inputs (or following light preprocessing), but use a metric that is

reflective of our perception of the data. Both approaches allow for data driven specifications and also direct incorporation of prior knowledge about the domain. We focus on the input space approach with the Wasserstein Ground Metric.

As presented in Sections 1.1,1.2 the Wasserstein distance is known to be an effective metric in the space of images, as demonstrated in image retrieval problems [RTG00, SDP15, SRG14] and related applications [PC19]. In particular, the Wasserstein distance is robust to natural variations such as translations and independent noise added to the pixel values. As observed with the Wasserstein Ground Metric, the Wasserstein-2 distance exhibits a Riemannian metric structure. In this section the Riemannian structure allows us to define an effective Wasserstein Gaussian noise[†] in the space of images which in expectation leads to a similar penalty term as in Section 1.4. The Wasserstein metric depends on the specific location at which it is being evaluated, and can define neighborhoods with a reasonable degree of semantic meaning. See, for example, the Wasserstein geodesics balls illustrated in Figure 1.1.

We suggest that regularization based on Wasserstein geometry can make a discriminative function noticeably smoother along the directions of natural variations of images, but without making it constant along the directions of semantic variation. Moreover, a generative perturbation model can be folded into the training objective (by computing the expectation value over perturbations of the Taylor expanded loss around each training example). This yields an effective penalty term that integrates (up to a given order in the expansion) a continuum of perturbations at once.

The rest of the section on Wasserstein adversarial robustness is organized as follows. In Subsection 1.5.2 we discuss relations of the proposed method to some of the existing literature. In Subsection 1.5.3 we review adversarial attacks in deep learning. In Subsection 1.5.4 we consider training with input noise and propose a Wasserstein Gaussian distribution in image space, which reflects the natural local variability of images. Then we compute the expectation

[†]Wasserstein Gaussians appear in the small time behavior of a process called Wasserstein diffusion, investigated in continuous [RS09] and discrete [Li18] states.

of the perturbed objective by Taylor expansions in the Wasserstein space. This leads to a Tikhonov-type Wasserstein diffusion smoothness regularizer. In Subsection 1.5.5 we present preliminary experimental results and in Subsection 1.5.6 we offer final remarks on adversarial regularization with the WGM.

1.5.2 Relevant literature to Wasserstein adversarial robustness

There are many works related to Wasserstein geometry, robustness, regularization. In this subsection we briefly mention some of the literature in relation to the focus of Section 1.5 on model robustness.

Adversarial robustness. Adversarial attacks are small-scale non-noticeable perturbations to the data inputs of neural networks that lead to large changes to the output and classification of discriminative models. Since adversarial attacks are problematic in many critical applications of deep learning, there has been a lot of work aiming to circumvent their effects in existing models. Previous works have investigated post-processing with Jacobian regularization [JG18] and cross Lipschitz regularization [HA17], whereby the input space was modeled with a Euclidean geometry space. The duality of attack norms and Lipschitz norms has been discussed as well in [FOA19]. Perturbation based regularization has been proposed in [YGZ18], which penalizes the negative effect of the adversarial attack in proportion to the size of the input. Gaussian data augmentation was proposed in [ZNR17] as well, but was instead evaluated by Monte Carlo samples and using Euclidean space. Recently, the trade-off between natural and robust classification errors was studied, leading to a training objective with an added term of the form $\mathbb{E}_{\mathbf{x}}[\max_{\mathbf{x}' \in B(\mathbf{x}, \epsilon)} \phi(f(\mathbf{x})f(\mathbf{x}')/\lambda)]$ [ZYJ19]. Similar to our method, this approach penalizes the variability of the classifier, but it does not incorporate priors about the geometry of the input data. Using the Wasserstein metric, [WSK19] uses modified Sinkhorn iterations to approximate projections of adversarial examples onto a Wasserstein ball. This is similar to the adversarial norm constraint that we present below.

However, our approach is based on a Riemannian metric formulation, which allows us to obtain a very simple quadratic form approximation of the norm and also enables us to integrate a generative noise model (Wasserstein diffusion) into an effective regularizer term added to the loss.

Robustness and regularization. Wasserstein balls have appeared in the context of robust density estimation [SKE17], where they are also related to a form of Tikhonov regularization in the case of logistic regression. Wasserstein distributionally robust stochastic optimization has been related to regularization by certain empirical gradient norms [GCK17]. The inspiration for this work, albeit not involving Wasserstein geometry, is the work by [Bis95], which shows that training with noise is equivalent to Tikhonov regularization.

1.5.3 Adversarial training and ground truth geometry

An adversarial attack is a perturbed version $\pi(\mathbf{x})$ of an input example \mathbf{x} , which alters the prediction of the classifier such that $f(\pi(\mathbf{x})) \neq f(\mathbf{x})$. According to this simplistic definition, every classifier can be successfully attacked, provided it has at least two possible output values. Taking a more refined perspective, consider $g(\mathbf{x})$ as the best possible classification, i.e., ground truth / Bayes classifier. Then a successful adversarial attack can be defined as a perturbation $\pi(\mathbf{x})$ of an input \mathbf{x} such that $f(\mathbf{x}) = g(\mathbf{x})$ but $f(\pi(\mathbf{x})) \neq g(\pi(\mathbf{x}))$. This highlights that what we care about is not whether a classifier changes its prediction when the input is perturbed, but rather in what scenarios does it change its prediction wrongfully, especially if it is a drastic change due to non-noticeable perturbation which will not be reflected by the ground truth g .

In order to quantify the sensitivity to attacks, we need a measure of the size of the perturbation model and the effect that it has on the classifier. Consider a loss function of the form

$$E(f) = \mathbb{E}_{p(y|\mathbf{x})p(\mathbf{x})} [l(f(\mathbf{x}), y)]. \quad (1.18)$$

We can measure the detriment of the loss when the data is perturbed in comparison with the unperturbed loss. For generality, we define a perturbation π at \mathbf{x} as a random variable. For example π can take the form of additive perturbations like $\pi(\mathbf{x}) = \mathbf{x} + \xi$, where ξ can be, e.g., a zero mean multivariate normal random variable, or a deterministic value obtained via an attack strategy on the input \mathbf{x} . The loss under perturbations is then

$$\mathbb{E}_{p(\pi|\mathbf{x})p(y|\pi(\mathbf{x}))p(\mathbf{x})} [l(f(\pi(\mathbf{x})), y)]. \quad (1.19)$$

As we noted earlier this will depend on the nature of the perturbations π . Adversarial examples are often constructed by minimizing the confidence of the discriminative function or increasing the training loss with respect to the input. Since this does not incorporate prior knowledge about the shape of the ground truth, usually the perturbations are restricted to lie within a very small L^p ball around the input example to ensure adversariality. When random input noise regularization and gradient penalties are applied using the same L^p losses, they suffer from generally regularizing in all directions, leading to a significant detriment in test accuracy. The situation is illustrated in the right part of Figure 1.1. Instead of restricting the perturbations to be small in an L^p norm sense, we suggest to refine the metric on input space and the perturbation model. For this we propose to measure distances on input space using the Wasserstein Ground Metric and train with a corresponding Wasserstein Gaussian input noise. The Wasserstein metric assigns a small distance to natural local variations of an input image. This means that larger perturbations are more likely to remain within the class of the input example that is being perturbed. In turn, we can apply higher levels of noise, allow for larger size perturbations in adversarial training, or apply stronger gradient penalties during training. In the next section we derive an effective regularizer for training with the Wasserstein metric on input space and which integrates the entire set of Wasserstein Gaussian noise perturbations (in a second order expansion) for each input example at once.

1.5.4 Perturbed loss and Wasserstein diffusion Tikhonov regularizer

It is well known that training with input noise can be related to training the original objective with an added penalty [Bis95]. These derivations usually are based on Taylor expansion of the perturbed loss around a given example. By default, the inputs are considered to live in Euclidean space, with loss functions such as the mean square error or the cross-entropy loss. Following the arguments from the previous sections, we model the input space of images as a Wasserstein space. We then derive the Wasserstein Taylor expansion of the perturbed loss and the corresponding regularization penalties. Once the input space is regarded as a Wasserstein metric space, our derivations follow Riemannian calculus reviewed in Section 1.2.

We consider a perturbation model defined in terms of a “Wasserstein Gaussian”, which at a given input image $\mathbf{x} \in \mathcal{X} = \mathcal{P}(\Omega)$ has a density function of the form

$$p(\xi|\mathbf{x}) = \exp(-d_{\mathcal{W}}^2(\mathbf{x}, \mathbf{x} + \xi)^2/\eta^2)d(\xi),$$

with a scale parameter $\eta > 0$ and a given reference measure $d(\xi)$. Locally, the Wasserstein-2 distance can be expressed as

$$d_{\mathcal{W}}(\mathbf{x}, \mathbf{x} + \xi)^2 = (\xi, G_{\mathcal{W}}(\mathbf{x})\xi)_{L^2} + o(\|\xi\|^2), \quad (1.20)$$

where $\xi \sim N(0, \eta^2\mathbf{I})$ and $G_{\mathcal{W}}(\mathbf{x})$ is the Wasserstein Riemannian metric tensor at \mathbf{x} , see (1.2.4). Note that unlike traditional Gaussian noise, ξ is dependent on the input image data \mathbf{x} , even with the linearization, as the covariance matrix $\eta^2 G_{\mathcal{W}}^{-1}(\mathbf{x})$ depends on the input \mathbf{x} . In addition, we note that $G_{\mathcal{W}}$ will depend on the choice of a ground metric d_{Ω} over pixels.

We are now ready to present our theorem that relates training with Wasserstein diffusion to training with an added penalty term.

Theorem 1.5.1 (Perturbed loss regularization). *Consider an input space (\mathcal{X}, g) with the Riemannian metric g represented by a matrix $G_{\mathcal{W}}(\mathbf{x})$ depending on $\mathbf{x} \in \mathcal{X}$, and consider the loss $E(f) = \mathbb{E}[l(f(\mathbf{x}), y)]$ from (1.18) with some error function l that is twice differentiable in the first argument. Let ξ be a Gaussian noise variable with zero mean and covariance*

matrix $\eta^2 G^{-1}(\mathbf{x})$ depending on \mathbf{x} . Then the perturbed loss from (1.19) takes the form

$$E_\xi(f) = E(f) + \frac{1}{2}\eta^2 E^R(f) + o(\eta^2),$$

where

$$E^R(f) = \mathbb{E}_{p(y|\mathbf{x})p(\mathbf{x})} \left[l''(f(\mathbf{x}), y) \|\nabla_{\mathbf{x}}^{\mathcal{W}^2} f(\mathbf{x})\|^2 + l'(f(\mathbf{x}), y) \Delta_{\mathcal{W}^2} f(\mathbf{x}) \right].$$

Here l' and l'' denote the first and second order ordinary partial derivatives of l with respect to the first argument, and ∇_g , $\|\cdot\|_g$, Δ_g are the gradient, norm, and Laplace-Beltrami operators on (\mathcal{X}, g) .

the intuition is that with the Wasserstein Ground Metric, the perturbation is proportional to the Riemannian steepest descent direction, $\xi \propto \nabla_{\mathbf{x}}^{\mathcal{W}^2} f(\mathbf{x}) = G^{-1}(\mathbf{x}) \nabla f(\mathbf{x})$, then the penalty is proportional to the Riemannian gradient norm squared, $\|\nabla_{\mathbf{x}}^{\mathcal{W}^2} f(\mathbf{x})\|^2$. This in contrast with the traditional isotropic Gaussian noise, where the Euclidean steepest descent is $\nabla_{\mathbf{x}} f(\mathbf{x})$.

Example 1 (Square error). For the square error $l(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$ and a perturbation model as in Theorem 1.5.1, we obtain the regularizer

$$E^R(f) = \mathbb{E}_{p(y|\mathbf{x})p(\mathbf{x})} \left[\|\nabla_{\mathbf{x}}^{\mathcal{W}^2} f(\mathbf{x})\|^2 + (f(\mathbf{x}) - y) \Delta_g f(\mathbf{x}) \right].$$

For non-zero mean perturbations, we can consider an expansion to first order which gives

$$E^R(f) = \mathbb{E}_{p(y|\mathbf{x})p(\mathbf{x})} \left[2(f(\mathbf{x}) - y) \cdot \mathbb{E}_{p(\xi|\mathbf{x})} [\xi]^\top \nabla f(\mathbf{x}) \right].$$

Example 2 (Cross entropy error). For the cross entropy $l(f(\mathbf{x}), y) = -y \ln(f(\mathbf{x})) - (1 - y) \ln(1 - f(\mathbf{x}))$ and a perturbation model as in Theorem 1.5.1, we obtain the regularizer

$$E^R(f) = \mathbb{E}_{p(y|\mathbf{x})p(\mathbf{x})} \left[\left(\frac{y}{f^2(\mathbf{x})} + \frac{1-y}{(1-f(\mathbf{x}))^2} \right) \|\nabla_{\mathbf{x}}^{\mathcal{W}^2} f(\mathbf{x})\|^2 + \left(-\frac{y}{f(\mathbf{x})} + \frac{(1-y)}{(1-f(\mathbf{x}))} \right) \Delta_g f(\mathbf{x}) \right].$$

In the case of k outputs (e.g., k -class classification), the loss function is summed for each output.

Example 3 (Euclidean inputs). *In the case of Euclidean inputs and uncorrelated zero mean isotropic Gaussian noise of variance η^2 , we recover some of the classic calculations by [Bis95]. Consider as an example the square error function, for which the regularizer becomes*

$$E^R(f) = \mathbb{E}_{p(y|x)p(\mathbf{x})} \left[\sum_i \left\{ \left(\frac{\partial f}{\partial \mathbf{x}_i} \right)^2 + (f(\mathbf{x}) - y) \frac{\partial^2 f}{\partial \mathbf{x}_i^2} \right\} \right].$$

As pointed out by [Bis95], this is the Tikhonov regularizer that is usually added to the sum of squares error.

Theorem 1.5.1 shows that all noise perturbed versions of a given input example can be integrated (in a second order sense) into a single term. Formally, equivalence of the regularizer to training with noise is only valid for small values of η , since it is based on a second order Taylor expansion. The Wasserstein diffusion smoothness regularizer E^R also has the natural interpretation as decreasing the variability of the classifier in an anisotropic and input dependent way that is captured by the Wasserstein gradient norm and the Laplace-Beltrami operator. This interpretation remains valid for arbitrarily large values of η , even if in this case the regularized objective might no longer correspond to the integrated perturbed objective.

We note that the term involving the Laplace-Beltrami operator is premultiplied with the derivative of the error. For regular choices of l , if the classifier makes correct predictions on the training inputs x (which is often the case), the derivative $l'(f(x), y)$ will vanish. This suggests that for the purpose of regularization in settings where the training error vanishes, in practice we can omit the Laplace-Beltrami term and consider only the gradient penalty term.

Taking the perspective of smoothness suggests that we may also regularize by penalizing the gradient of the discriminator, instead of the gradient of the loss function. Finally, we point out that the Wasserstein metric can also be used to define the size constraints for adversarial training. Usually adversarial perturbations are constrained to have L^∞ norm (or some L^p norm) bounded by a small ϵ . Instead of using $\|\xi\|_{L^p} \leq \epsilon$, we can use $\|\xi\|_{\mathcal{W}^2} \leq \epsilon$, or simply $\xi^\top G_{\mathcal{W}^2}(x)\xi \leq \epsilon$.

1.5.5 Experiments

Below we present experimental results to evaluate the utility of Wasserstein smoothness regularization in terms of the robustness of the trained classifiers to small and large perturbations. We focus on regularization by the Wasserstein gradient norm penalty. For the Wasserstein Ground Metric regularization, for each training example \mathbf{x} we add

$$l''(f(\mathbf{x}), y) \cdot \|\nabla_{\mathbf{x}}^{\mathcal{W}^2} f(\mathbf{x})\|^2. \quad (1.21)$$

1.5.5.1 Stability to adversarial perturbations of the input data

In this experiment we test the effectiveness of the gradient penalty regularizer in terms of the test accuracy of the trained classifiers. We train a ResNet-20 on clean images from CIFAR-10 with gradient norm penalty computed under Euclidean and Wasserstein metrics. We run grid search for the regularization strength and the radius defining the ground metric on pixel space. The training error converges to zero in all cases. We consider two types of test data: the clean test dataset (natural generalization) and the test dataset with each test example perturbed by an adversarial attack (robust generalization). Following current literature, adversarial perturbations are computed by FGSM and I-FGSM [GSS15, KGB17]. More details on the implementation and hyperparameters are provided in Appendix 1.A.3. Our results, reported in Table 1.2, compare with the Fast Gradient Sign Method (FGSM) [GSS15] and the iterative counterpart, I-FGSM [KGB17].

Test data / Regularizer	None	Euclidean grad.	Wassserstein grad.
Natural	16.29	15.61	15.35
FGSM $\epsilon = 8/255$	82.22	31.10	30.20
FGSM $\epsilon = 25/255$	89.72	66.83	44.32
I-FGSM-20 $\alpha = 2/255, \epsilon = 8/255$	90.15	40.06	32.12

Table 1.2: Robust test error percentage (lower is better) for a ResNet-20 network with softplus activation trained for 200 epochs on clean CIFAR-10 training images using gradient norm regularization with Euclidean and Wasserstein metric on image space. We run grid search over the regularization strength and the ground metric radius on pixel space.

1.5.5.2 Stability to large in-class variations of the input data

Most work on adversarial robustness focuses on small perturbations, with adversarial attacks restricted to have a small norm so that they remain imperceptible to humans. We are interested in generalization for all kinds of in-class variations of the data, including large perturbations that should not change the predicted class. In this experiment we train on the clean CIFAR-10 training set (no data augmentation), and compare between no regularization, Euclidean, and Wasserstein smoothness regularization. For testing, we randomly draw 1000 images from the test set and construct for each of them a sequence of translated versions with padding, as depicted in Figure 1.8. The semantic meaning of images should remain relatively constant under incremental translations, and therefore we expect a robust classifier to label all images in the sequence similarly. Quantitatively, this is measured by the number of label flips that occur in the sequence. We report the average number of label flips over all sequences of test images in Table 1.3. As the table shows, Wasserstein gradient regularization improves the robustness of the classifier to translations.



Figure 1.8: Robust classifiers should be invariant to natural variations of the data. Shown are horizontal translations of an image from CIFAR-10.

Perturbation \ Regularizer	None	Euclidean grad.	Wasserstein grad.
Horizontal translation	10.009	7.898	6.488
Vertical translation	9.920	9.437	7.956

Table 1.3: Average number of prediction flips on sequences of translated test images from CIFAR-10. The classifiers were trained on the clean CIFAR-10 training set with no data augmentation, with either no regularization, Euclidean Tikhonov regularization, or Wasserstein Tikhonov regularization.

1.5.6 Discussion

Training with input noise or data augmentation in general is known as an effective form of regularization to obtain classifiers that are more robust to natural variations of the data, or to reduce the sensitivity to perturbations. These methods usually have a high cost in terms of the number of examples needed and the cost of computing each of them (especially in the case of adversarial data augmentation obtained by iterated gradient methods). Another problem is that usually noise models and adversarial examples need to be restricted to tiny norm values to ensure that they remain within the class of the perturbed example. Smoothness regularizers based on L^p metrics are usually limited in the same way. This section follows the idea that the space of inputs is not Euclidean and that smoothness priors should be implemented with respect to an appropriate metric, which in turn would allow us to apply higher levels of regularization without hurting test performance. We propose to use the Wasserstein-2 metric to capture semantically meaningful neighborhoods of images. As we

show, the Wasserstein diffusion smoothness regularizer arises naturally by expanding and integrating the loss with respect to Wasserstein Gaussian noise on the inputs. We obtain an effective penalty that can be computed very efficiently, saving computation compared with adversarial data augmentation, and has a negligible overhead over L^2 gradient penalties. The experimental results indicate that our methods can improve robust generalization performance on CIFAR-10 for both adversarial robustness and natural variations.

1.A Appendix

1.A.1 Proof of equivalence of noise training with Wasserstein Thikonov Regularization

A Riemannian metric g defines an inner product between tangent vectors of the input space at each possible location. We choose standard coordinates for the input space $\mathcal{X} = \mathbb{R}^n$ and write $g(\xi, \zeta) = (\xi, \zeta)_g = \xi^\top G(x)\zeta$ for any pair $\xi, \zeta \in T_x\mathcal{X}$. We implicitly identify $T_x\mathcal{X}$ and \mathcal{X} so that adding a tangent vector $\xi \in T_x\mathcal{X}$ to an input vector $x \in \mathcal{X}$ makes sense. The Riemannian gradient with the metric g is given by $\nabla_g f(x) = G^{-1}(x)\nabla f(x)$, where ∇ is the ordinary gradient. This is also known as the natural gradient.

Proof of Theorem 1.5.1:

We expand the error function l around a data point x with added noise ξ in the Riemannian space (\mathcal{X}, g) . We obtain

$$\begin{aligned} l(f(x + \xi), y) &= l(f(x), y) + l'(f(x), y)(\nabla_g f(x), \xi)_g \\ &\quad + \frac{1}{2}l''(f(x), y)(\nabla_g f(x), \xi)_g^2 + \frac{1}{2}l'(f(x), y) \sum_{i,j} \xi_i \xi_j (\nabla_g^2 f(x))_{ij} + o(\|\xi\|_g^2). \end{aligned}$$

We discuss the individual terms in turn. The zero order term is just the unperturbed loss. On taking the expectation value with respect to ξ given x , the linear term vanishes when we assume that the perturbations have zero mean, $\mathbb{E}_{p(\xi|x)}[\xi] = 0$. If the perturbation does not have zero mean, we obtain

$$\mathbb{E}_{p(\xi|x)}[(\nabla_g f(x), \xi)_g] = \mathbb{E}_{p(\xi|x)}[(G^{-1}(x)\nabla f(x))^\top G(x)\xi] = \nabla f(x)^\top \mathbb{E}_{p(\xi|x)}[\xi].$$

For the first quadratic term, when $\mathbb{E}_{p(\xi|x)}[\xi\xi^\top] = \eta^2 G^{-1}(x)$, we obtain

$$\mathbb{E}_{p(\xi|x)}[(\nabla_g f(x)^\top G(x)\xi)^2] = \eta^2 \nabla_g f(x)^\top G(x)\nabla_g f(x) = \eta^2 \|\nabla_g f\|_g^2.$$

For the second quadratic term, again when $\mathbb{E}_{p(\xi|x)}[\xi\xi^\top] = \eta^2 G^{-1}(x)$, we obtain

$$\mathbb{E}_{p(\xi|x)}[\xi^\top \text{Hess } f(x)\xi] = \eta^2 \Delta_g f(x).$$

Here the Laplace-Beltrami operator is

$$\Delta_g f = \sum_{j,k} g^{jk} \frac{\partial^2 f}{\partial x^j \partial x^k} - g^{jk} \Gamma_{jk}^l \frac{\partial f}{\partial x^l},$$

where Γ_{jk}^l is the Christoffel symbol. □

1.A.2 Wasserstein metric in un-normalized distributions

We illustrate the Wasserstein metric tensor in un-normalized density space. The new metric tensor induces the gradient operator in un-normalized density space. This follows the recent work on the topic including [GLO19].

In other words, consider

$$\mathcal{M}_+(I) = \left\{ \mu = (\mu_1, \dots, \mu_n) \in \mathbb{R}^n : \mu_i \geq 0 \right\}.$$

The tangent space of $\mathcal{M}_+(I)$ at μ forms

$$T_\mu \mathcal{M}_+(I) = \mathbb{R}^n.$$

Definition 1.A.1 (Unnormalized Wasserstein-2 metric tensor). *The inner product $\tilde{g}_\mu : T_\mu \mathcal{M}_+(I) \times T_\mu \mathcal{M}_+(I) \rightarrow \mathbb{R}$ forms*

$$\tilde{g}_\mu(\sigma_1, \sigma_2) := \sigma_1^\top \left(L(p)^\dagger + \frac{1}{\alpha} \mathbf{1}\mathbf{1}^\top \right) \sigma_2,$$

for any $\sigma_1, \sigma_2 \in T_p \mathcal{P}_+(I)$.

It is clear that $(\mathcal{M}_+(I), \tilde{g})$ is a well defined metric in positive octant. In this case, the un-normalized Wasserstein-2 gradient is given by the following theorem.

Theorem 1.A.1 (Unnormalized Wasserstein-2 gradient on graphs). *Given $\mathcal{F} \in C^1(\mathcal{M}_+(I))$, the gradient operator in Riemannian manifold $(\mathcal{M}_+(I), \tilde{g})$ satisfies*

$$\text{grad } \mathcal{F}(\mu) = \left(L(\mu) + \alpha \mathbf{1}\mathbf{1}^\top \right) d_\mu \mathcal{F}(\mu).$$

In other words,

$$\begin{aligned} \text{grad } \mathcal{F}(\mu)_i &= \frac{1}{2} \sum_{j \in N(i)} \omega_{ij} \left(\frac{\partial}{\partial \mu_i} \mathcal{F} - \frac{\partial}{\partial \mu_j} \mathcal{F} \right) \left(\frac{\mu_i}{d_i} + \frac{\mu_j}{d_j} \right) \\ &\quad + \alpha \sum_{i=1}^n \frac{\partial}{\partial \mu_i} \mathcal{F}(\mu). \end{aligned}$$

Proof: Notice that

$$L(\mu) = T \begin{pmatrix} 0 & & & \\ & \lambda_{\text{sec}}(L(\mu)) & & \\ & & \ddots & \\ & & & \lambda_{\text{max}}(L(\mu)) \end{pmatrix} T^{-1},$$

where $0 < \lambda_{\text{sec}}(L(\mu)) \leq \dots \leq \lambda_{\text{max}}(L(\mu))$ are eigenvalues of $L(\rho)$ arranged in ascending order, and T is its corresponding eigenvector matrix. Here the zero eigenvalue correspond to the eigenvector $\mathbf{1}$. Thus

$$\left(L(\mu)^{-1} + \frac{1}{\alpha} \mathbf{1}\mathbf{1}^T \right)^{-1} = L(\mu) + \alpha \mathbf{1}\mathbf{1}^T.$$

Then

$$\begin{aligned} \text{grad } \mathcal{F}(\mu) &= \left(L(\mu)^{-1} + \frac{1}{\alpha} \mathbf{1}\mathbf{1}^\dagger \right)^{-1} d_\mu \mathcal{F}(\mu) \\ &= L(\mu) d_\mu \mathcal{F}(\mu) + \alpha \mathbf{1}\mathbf{1}^T d_\mu \mathcal{F}(\mu), \end{aligned}$$

which finishes the proof. □

1.A.3 Detailed description of the experiments

Image generation experiments We run experiments on the CIFAR-10 and CelebA (aligned, cropped, 64×64) datasets.

For the experiment measuring discriminator robustness to noise, or hyperparameters for WGAN-GP is,

- DCGAN Architecture, with 3 convolutional layers, and no batch-normalization in the discriminator.

- Adam optimizer, with learning rate 0.0003, and $\beta_1 = 0.5$, and $\beta_2 = 0.9$
- Batch size of 64, and noise vector of dimension 128.

For the WWGAN loss, we use the same hyperparameters as WGAN-GP, and for the WWGAN, we set $\alpha = 1.0$ and $\beta = 50$.

For the noise model, we used RGB salt and pepper noise, which first transforms the $3 \times N \times N$ image to a $3N^2$ vector, and provides a probability of changing any coordinate. Once a change is decided, the coordinate value is set to 0.0 or 1.0 (the max pixel value) with equal probability.

Then the discriminator is evaluated on 64 noisy and clean images. And we see that the discriminator trained with WWGAN is more robust to noise.

We compare the WWGAN loss function with the WGAN-GP loss. For both losses, we use a DCGAN architecture, removing the batch-normalization layer in the discriminator. We also train with the Adam optimizer with learning rate $1e - 4$ and $\beta_1 = 0.9$, $\beta_2 = 0$.

Codebase for WWGAN The WWGAN algorithm is available in Github at this link <https://github.com/duklerioni/WWGAN>

Wasserstein diffusion for adversarial robustness For our experiments, we use the CIFAR-10 dataset, and use white-box attacks on the ResNet-20 network. For training, we fixed the batch size of 128, and used SGD with momentum and weight decay, where the momentum value is 0.9 and the weight-decay value is 10^{-4} . We start with a learning rate of 0.1, and at epoch 100 and 150 we divide the learning rate by 10 each time. Our activations were softplus with a $\beta = 1$ and a threshold of 20, where the softplus activation is

$$\text{Softplus}(x) = \frac{1}{\beta} \log(1 + \exp(\beta x)),$$

and the threshold value is the value of x beyond which we assume that the softplus equals a linear function, for numerical stability.

We examine the case of training the ResNet-20 network on the CIFAR-10 dataset, with standard normalization of the pixel values. This achieves a test accuracy of 83.71%. We then examine the effect of modifying the loss objective with either the Euclidean or Wasserstein gradient penalties of the original loss, namely we use the loss

$$\ell(f(x), y) + \eta^2(\nabla_x \ell(f(x), y), G(x)^{-1} \nabla_x \ell(f(x), y)),$$

where ∇_x is the Euclidean gradient and $G(x) \in \mathbb{R}^{d \times d}$ represents the metric used in sample space. For the Wasserstein gradient norm, $G(x)^{-1} = L(x)$. For the Euclidean gradient norm, $G(x) = I$.

The values of the regularization strength η^2 we tested were

$$10^{-3}, 10^{-2}, 0.1, 1, 10, 100, 10^3, 10^4, 10^5. \tag{1.22}$$

And for the ground metric on pixel space we considered neighbors in a square shape, where the size of the square had half the side-length be 2, 4, 6, 8.

1.A.4 WWGAN generated images

Figures 1.9 and 1.10 below show sample images generated from the WWGAN model trained with the settings described in Appendix 1.A.3.

Fake Images

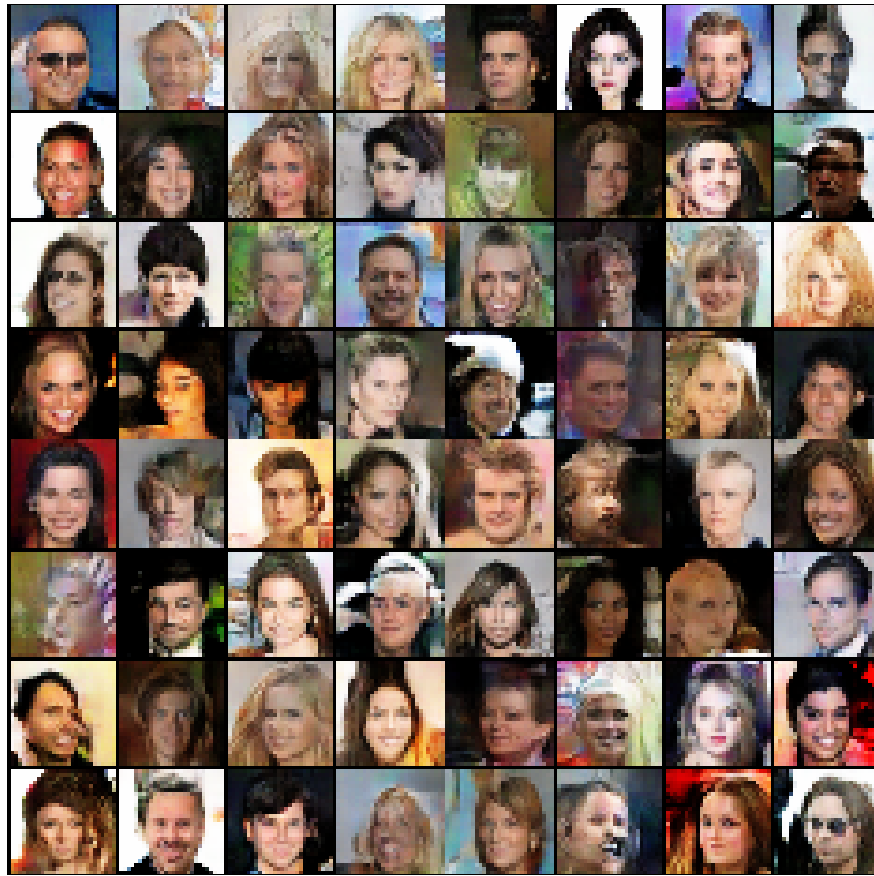


Figure 1.9: CelebA cropped 64×64 WGAN generated images.



Figure 1.10: CIFAR-10 WGAN generated images.

CHAPTER 2

Differentiable dataset optimization^{*†}

2.1 Introduction

Consider the following seemingly disparate questions. (i) *Dataset Extension*: Given a relatively small training set, but access to a large pool of additional data, how to select from the latter samples to augment the former? (ii) *Dataset Curation*: Given a potentially large dataset riddled with annotation errors, how to automatically reject such outlier samples? (iii) *Dataset Reweighting*: Given a finite training set, how to reweight the training samples to yield better generalization performance?

These three are examples of *Dataset Optimization*. In order to solve this problem with differentiable programming, one can optimize a loss of the model end-to-end, which requires *differentiating the model's loss with respect to the dataset*. Our main contribution is an efficient method to compute such a *dataset derivative*. This allows learning an importance weight α_i for each datum in a training dataset \mathcal{D} , extending the optimization from the weights \mathbf{w} of a parametric model such as a deep neural network (DNN), to also include the weights of the dataset.

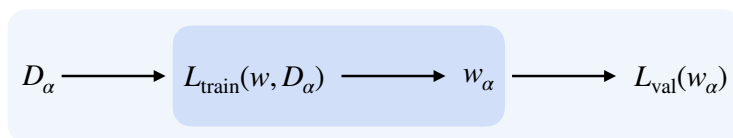
As illustrated in the following diagram, standard optimization in machine learning works

^{*}This Chapter is adapted from [Ano22] and is the result of work done during an internship at Amazon Research.

Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved for work in this chapter.

[†]This chapter reproduces material from [Ano22], with the permission from coauthors

by finding the weights \mathbf{w}_α that minimize the training loss $L_{\text{train}}(\mathbf{w}, D_\alpha) = \sum_i \alpha_i \ell(f_{\mathbf{w}}(\mathbf{x}_i), y_i)$ on a given (weighted) dataset D_α (dark box). We solve a more general learning problem (light box) by jointly optimizing the dataset \mathcal{D}_α in addition to \mathbf{w} . To avoid the trivial solution $\alpha = 0$, it is customary in AutoML to optimize \mathcal{D}_α by minimizing the validation error computed on a disjoint dataset. This makes for inefficient use of the data, which has to be split between training and validation sets. Instead, we leverage a closed-form expression of the leave-one-out cross-validation error to jointly optimize the model and data weights during training, without the need to create a separate validation set.



The intermediate block in the diagram (which finds the optimal weights \mathbf{w}_α for the the training loss on \mathcal{D}_α) is usually non-differentiable with respect to the dataset, or the derivative is prohibitively expensive to compute. DIVA leverages recent progress in deep learning linearization [AGR20], to derive a closed-form expression for the derivative of the final loss (validation error) with respect to the dataset weights. In particular, [AGR20] have shown that, by replacing cross-entropy with least-squares, replacing ReLu with leaky-ReLu, and performing suitable pre-conditioning, the linearized model performs on par with full non-linear fine-tuning. We also leverage a classical result to compute the leave-one-out loss of a linear model in closed-form [RL07, GS93]. This allows us to optimize the LOO loss without requiring a separate validation set, setting DIVA apart from bi-level optimization customary in AutoML.

To illustrate the many possible uses of the dataset derivative, we run experiments with a simplified version of DIVA to cleanup a dataset of noisy annotations, to extend a training set with additional data from an external pool, identify meaningful data augmentation, and to perform multi-modal expansion using a CLIP model [RKH21].

Rather than using the full linearization of the model derived by [AGR20], we restrict the

gradient to its last layer, cognizant that we are not exploiting the full power of LQF and thereby obtaining only a lower-bound of performance improvement. Despite that restriction, our results show consistent improvements from dataset optimization, at the modest computational cost of a forward pass over the dataset to optimize the importance weights.

To summarize, our main contributions are:

1. We introduce a method to compute the dataset derivative in closed form, DIVA.
2. We illustrate the use of DIVA to perform dataset optimization by minimizing directly the leave-one-out error without the need for an explicit validation dataset.
3. We perform experiments with a simplified model that, despite not using the full power of the linearization, show consistent improvements in dataset extension, re-weighting, outlier rejection and automatic aggregation of multi-modal data.

Our method presents several limitations. The dataset derivative of a learning task is computed around a point represented by a pre-trained model. It only allows local optimization around this point. Moreover, we only compute a restriction of the linearization to the dimensions spanned by the last few layers. In general, this yields suboptimal results compared to full global optimization from scratch, if one could compute that at scale. Nonetheless, the linearized setting is consistent with the practice of fine-tuning pre-trained models in light of the results of [AGR20], see also [RKH21, RRC17, MGM18, HKF18].

2.2 Related work

AutoML. State of the art performance in image classification tasks often relies on large amount of human expertise in selecting models and adjusting the training settings for the task at hand [LCY20]. Automatic machine learning (AutoML) [FKE19, HZC21] aims to automate model selection [CT10] and the training settings by instead using meta-algorithms for the different aspects of the learning settings. Such methods follow a bi-level optimization framework, optimizing the training settings in the outer level, and traditional model optimization in the

inner level [JF18]. AutoML has focused on achieving better results via automatic model selection [DAR21, FKE19] including neural architecture search (NAS) [ZL16, EMH19, LCS19]. Other important AutoML topics include hyper-parameter selection [LJD17, ASY19] and data augmentation [CZM18, LKK19, CCC20, BBG20], which are closer to our settings of optimizing the dataset weights. Since the main signal for a model’s performance is the final validation loss, which requires full optimization of the model for each evaluation, AutoML approaches often incur a steep computational costs. Alternatively, other methods follow alternating optimization of the criteria, such as the work of [RZY18] that approximates full network optimization with a single SGD step to learn to reweight the training set dynamically. *Differentiable AutoML* alleviates outer-optimization costs while optimizing the final validation error via differentiable programming, by utilizing proxy losses and continuous relaxation that enable differentiation. Different approaches to differentiable AutoML include differentiable NAS [LSY18, WZ19], data augmentation [LHH21, LHW20], and hyper-parameter optimization [ADG16]. The DIVA dataset derivative follows the differentiable AutoML framework by enabling direct optimization of the dataset with respect to the final validation error of the model.

Importance sampling. While our dataset optimization problem may seem superficially similar to importance sampling, the optimization objective is different. Importance sampling aims to reweight the training set to make it more similar to the test distribution or to speed up convergence. On the other hand, DIVA’s objective is to optimize a validation loss of the model, even if this requires making the training distribution significantly different from the testing distribution. Importance sampling methods have a long history in the MCMC machine learning literature where the sampling is conditioned on the predicted importance of samples [MU49, Liu08]. In deep learning, importance sampling methods have been studied theoretically for linearly-separable data [BL19] and recently in more generality [XYR21]. Furthermore, there exist many importance sampling heuristics in deep learning training including different forms of hard sample mining [SGG16, XHZ19, CLM17], weighting based

on a focal loss [LGG17], re-weighting for imbalance, [CJL19, HLL19, DGZ17] and gradient based scoring [LLW19]. We emphasize that DIVA’s optimization of the sample weights is not based on a heuristic but is rather a differentiable AutoML method driven by optimization of a proxy of the test error. Further, DIVA allows optimization of the dataset weights with respect to an arbitrary loss and also allows for dataset extension computation.

LOO based optimization. Leave-one-out cross validation is well established in statistical learning [Sto77]. In ridge regression, the LOO model predictions for the validation samples have a closed-form expression that avoids explicit cross validation computation [GS93, RL07] enabling efficient and scalable unbiased estimate of the test error. Efficient LOO has been widely used as a criterion for regularization [PVG11, QLL10, BBB99, TMW20], hyperparameter selection [HS17] and optimization [WHY08]. Most similar to our dataset derivative are methods that: (1) optimize a restricted set of parameters, such as kernel bandwidth, in weighted least squares [Caw06, HCH07] (2) locally weighted regression methods (*memorizing regression*) [AMS97, MHJ92], or (3) methods that measure the impact of samples based on LOO predictions [BF99, NRS21].

Dataset selection & sample impact measures. [KL17] measure the effect of changes of a training sample weight on a final validation loss through per-sample weight gradients, albeit without optimizing the dataset and requiring a separate validation set. Their proposed expression for the per-sample gradient, however, does not scale easily to our problem of dataset optimization. In contrast, in proposition 5 we introduce an efficient closed-form expression for the derivative of the whole datasets. Moreover, in proposition 5, we show how to optimize the weights with respect to a cross-validation loss which does not require a separate set.

In [PLS20], the authors present a sample-impact measure for interpretability based on a validation set; for dataset extension, [YAF20] presents a coarse dataset extension method based on self-supervised learning. Dataset distillation and core set selection methods aim

to decrease the size of the dataset [WZT18] by selecting a representative dataset subset [HJS20, JHS20, CYM19, JVE20, TB18, KSM21]. While DIVA is capable of removing outliers, in this work we do not approach dataset selection from the perspective of making the dataset more computationally tractable by reducing the number of samples.

2.3 Proposed method

In supervised learning, we use a parametrized model $f_{\mathbf{w}}(\mathbf{x})$ to predict a target output y given an input \mathbf{x} coming from a joint distribution $(\mathbf{x}, y) \sim \mathcal{T}$. Usually, we are given a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ with samples (\mathbf{x}, y) assumed to be independent and identically distributed (i.i.d.) according to \mathcal{T} . The training set \mathcal{D} is then used to assemble the empirical risk for some per-sample loss ℓ ,

$$L_{\text{train}}(\mathbf{w}; \mathcal{D}) = \sum_{i=1}^N \ell(f_{\mathbf{w}}(\mathbf{x}_i), y_i),$$

which is minimized to find the optimal model parameters $\mathbf{w}_{\mathcal{D}}$:

$$\mathbf{w}_{\mathcal{D}} = \underset{\mathbf{w}}{\operatorname{argmin}} L_{\text{train}}(\mathbf{w}; \mathcal{D}).$$

The end goal of empirical risk minimization is that the weights will also minimize the *test loss*, computed using a separate test set. Nonetheless \mathcal{D} is often biased and differs from the distribution \mathcal{T} . In addition, from the perspective of optimization, different weighting of the training loss samples can enable or inhibit good learning outcomes of the task \mathcal{T} [LGG17].

Dataset Optimization. In particular, it may not be the case that sampling the training set \mathcal{D} i.i.d. from \mathcal{T} is the best option to guarantee generalization, nor it is realistic to assume that \mathcal{D} is a fair sample. Including in-distribution samples that are too difficult may negatively impact the optimization, while including certain out-of-distribution examples may aid the generalization on \mathcal{T} . It is not uncommon, for example, to improve generalization by training on a larger dataset containing out-of-distribution samples coming from other sources, or generating out-of-distribution samples with data augmentation. We call Dataset Optimization

the problem of finding the optimal subset of samples, real or synthetic, to include or exclude from a training set \mathcal{D} in order to guarantee that the weights $\mathbf{w}_{\mathcal{D}}$ trained on \mathcal{D} will generalize as much as possible.

Differentiable Dataset Optimization. Unfortunately, a naïve brute-force search over the 2^N possible subsets of \mathcal{D} is unfeasible. The starting idea of DIVA is to instead solve a more general continuous optimization problem that can be optimized end-to-end. Specifically, we parameterize the choice of samples in the augmented dataset through a set of non-negative continuous sample weights α_i which can be optimized by gradient descent along with the weights of the model. Let $\alpha = (\alpha_1, \dots, \alpha_N)$ be the vector of the sample weights and denote the corresponding weighted dataset by \mathcal{D}_α . The training loss on \mathcal{D}_α is then defined as:

$$L_{\text{train}}(\mathbf{w}; \mathcal{D}_\alpha) = \sum_{i=1}^N \alpha_i \ell(f_{\mathbf{w}}(\mathbf{x}_i), y_i). \quad (2.1)$$

Note that if all α_i 's are either 0 or 1, we are effectively selecting only a subset of \mathcal{D} for training. As we will show, this continuous generalization allows us to optimize the sample selection in a differentiable way. In principle, we would like to find the sample weights $\alpha^* = \operatorname{argmin}_\alpha L_{\text{test}}(\mathbf{w}_\alpha)$ that lead to the best generalization. Since we do not have access to the test data, in practice this translates to optimizing α with respect to an (unweighted) validation loss L_{val} :

$$\alpha^* = \operatorname{argmin}_\alpha L_{\text{val}}(\mathbf{w}_\alpha).$$

We can, of course, compute a validation loss using a separate validation set. However, as we will see in Section 2.3.3, we can also use a leave-one-out cross-validation loss directly on the training set, without any requirement of a separate validation set.

In order to efficiently optimize α by gradient-descent, we need to compute the dataset derivative $\nabla_\alpha L_{\text{val}}(\mathbf{w}_\alpha)$. By the chain rule, this can be done by computing $\nabla_\alpha \mathbf{w}_\alpha$. However, the training function $\alpha \rightarrow \mathbf{w}_\alpha$ that finds the optimal weights \mathbf{w}_α of the model given the sample weights α may be non-trivial to differentiate or may not be differentiable at all (for example,

it may consist of thousands of steps of SGD). This would prevent us from minimizing α end-to-end.

In the next section, we show that if, instead of linearizing the \mathbf{w}_α end-to-end in order to compute the derivative, we linearize the model *before* the optimization step, the derivative can both be written in closed-form and computed efficiently, thus giving us a tractable way to optimize α .

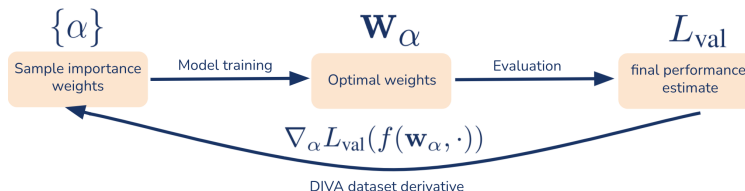


Figure 2.1: The DIVA dataset derivative is computed end-to-end from the final validation loss

2.3.1 Linearization

In real-world applications, the parametric model $f_{\mathbf{w}}(\mathbf{x})$ is usually a deep neural network. Recent work [AGR20, MLL20] have shown that in many cases, a deep neural network can be transformed to an equivalent linear model that can be trained on a simple quadratic loss and still reach a performance similar to the original model. Given a model $f_{\mathbf{w}}(\mathbf{x})$, let \mathbf{w}_0 denote an initial set of weights, For example, \mathbf{w}_0 could be obtained by pre-training on a large dataset such as ImageNet (if the task is image classification). Following [AGR20, MLL20], we consider a linearization $f_{\mathbf{w}}^{\text{lin}}(\mathbf{x})$ of the network $f_{\mathbf{w}}(\mathbf{x})$ given by the first-order Taylor expansion of $f_{\mathbf{w}}(\mathbf{x})$ around \mathbf{w}_0 :

$$f_{\mathbf{w}}^{\text{lin}}(\mathbf{x}) = f_{\mathbf{w}_0}(\mathbf{x}) + \nabla_{\mathbf{w}} f_{\mathbf{w}_0}(\mathbf{x}) \cdot (\mathbf{w} - \mathbf{w}_0). \quad (2.2)$$

Intuitively, if fine-tuning does not move the weights much from the initial pre-trained weights \mathbf{w}_0 , then $f_{\mathbf{w}}^{\text{lin}}(\mathbf{x})$ will remain a good approximation of the network while becoming linear in \mathbf{w} (but still remaining highly non-linear with respect to the input \mathbf{x}). Effectively, this

is equivalent to training a linear classifier using the gradients $\mathbf{z}_i := \nabla_{\mathbf{w}} f_{\mathbf{w}_0}(\mathbf{x}_i)$ as features [MLL20].

Although $f_{\mathbf{w}}^{\text{lin}}(\mathbf{x})$ is a linear model, the optimal weights \mathbf{w}_α may still be a complex function of the training data, depending on the loss function used. [AGR20] showed that equivalent performance can be obtained by replacing the empirical cross-entropy with the regularized least-squares loss:

$$L_{\text{train}}(\mathbf{w}) = \sum_{i=1}^N \|f_{\mathbf{w}}^{\text{lin}}(\mathbf{x}) - \mathbf{y}_i\|^2 + \lambda \|\mathbf{w}\|^2 \quad (2.3)$$

where \mathbf{y} denotes the one-hot encoding vector of the label y_i . In [AGR20], it is shown that linearized models are equivalent from the standpoint of performance on most standard tasks and classification benchmarks, and better in the low-data regime, which is where the problem of “dataset augmentation” is most relevant. The advantage of using this loss is that the optimal weights \mathbf{w}^* can now be written in closed-form as

$$\mathbf{w}^* = (\mathbf{Z}^\top \mathbf{Z} + \lambda \mathbf{I})^{-1} \mathbf{Z}^\top (\mathbf{Y} - f_{\mathbf{w}_0}(\mathbf{X})), \quad (2.4)$$

where $\mathbf{Z} = [\mathbf{z}_1, \dots, \mathbf{z}_N]$ is the matrix of the Jacobians $\mathbf{z}_i = \nabla_{\mathbf{w}} f_{\mathbf{w}_0}(\mathbf{x}_i)$. While our method can be applied with no changes to linearization of the full network, for simplicity in our experiments we restrict to linearizing only the last layer of the network. This is equivalent to using the network as a fixed feature extractor and training a linear classifier on top the last-layer features, that is, $\mathbf{z}_i = f_{\mathbf{w}_0}^{L-1}(\mathbf{x}_i)$ are the features at the penultimate layer.

2.3.2 Computation of the dataset derivative

We now show that for linearized models we can compute the derivative $\nabla_{\alpha} \mathbf{w}_\alpha$ in closed-form. For the α -weighted dataset, the objective in (2.3) with L_2 loss for the linearized model is written as,

$$\mathbf{w}_\alpha = \underset{\mathbf{w}}{\text{argmin}} L_{\text{train}}(\mathbf{w}; \mathcal{D}_\alpha) = \underset{\mathbf{w}}{\text{argmin}} \sum_{i=1}^N \alpha_i \|\mathbf{w}^\top \mathbf{z}_i - \mathbf{y}_i\|^2 + \lambda \|\mathbf{w}\|^2. \quad (2.5)$$

where $\mathbf{z}_i = \nabla_{\mathbf{w}} f_{\mathbf{w}_0}(\mathbf{x}_i)$ as in the previous section. Note that $\alpha_i \|\mathbf{w}^\top \mathbf{z}_i - \mathbf{y}_i\|^2 = \|\mathbf{w}^\top \mathbf{z}_i^\alpha - \mathbf{y}_i^\alpha\|^2$, where $\mathbf{z}_i^\alpha := \sqrt{\alpha_i} \mathbf{z}_i$ and $\mathbf{y}_i^\alpha := \sqrt{\alpha_i} \mathbf{y}_i$. Using this, we can reuse (2.4) to obtain the following

closed-form solution for \mathbf{w}_α :

$$\mathbf{w}_\alpha = (\mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Z} + \lambda \mathbf{I})^{-1} \mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Y}, \quad (2.6)$$

where we have taken $\mathbf{D}_\alpha = \text{diag}(\alpha)$. In particular, note that \mathbf{w}_α is now a differentiable function of α . The following proposition gives a closed-form expression for the derivative.

Proposition 3 (Model-Dataset Derivative $\nabla_\alpha \mathbf{w}_\alpha$). *For the ridge regression problem (2.5) and \mathbf{w}_α defined as in (2.6), define*

$$\mathbf{C}_\alpha = (\mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Z} + \lambda \mathbf{I})^{-1}. \quad (2.7)$$

Then the Jacobian of \mathbf{w}_α with respect to α is given by

$$\nabla_\alpha \mathbf{w}_\alpha = \mathbf{Z} \mathbf{C}_\alpha \circ ((\mathbf{I} - \mathbf{Z} \mathbf{C}_\alpha \mathbf{Z}^\top \mathbf{D}_\alpha) \mathbf{Y}), \quad (2.8)$$

Where we write $\mathbf{A} \circ \mathbf{B} \in \mathbb{R}^{n \times m \times k}$ for the batch-wise outer product of $\mathbf{A} \in \mathbb{R}^{n \times m}$ and $\mathbf{B} \in \mathbb{R}^{n \times k}$ along the common dimension k , i.e., $(\mathbf{A} \circ \mathbf{B})_{ijk} = a_{ij} b_{ik}$.

The Jacobian $\nabla_\alpha \mathbf{w}_\alpha$ would be rather large to compute explicitly. Fortunately, the end-to-end gradient of the final validation loss, $L_{\text{val}}(\mathbf{w}_\alpha)$, can still be computed efficiently, as we now show. Given a validation dataset \mathcal{D}_{val} , the validation loss is:

$$L_{\text{val}}(\mathbf{w}_\alpha) = \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_{\text{val}}} \ell(f_{\mathbf{w}_\alpha}(\mathbf{x}_i), y_i). \quad (2.9)$$

The following gives the expression from which we optimize α end-to-end with respect to the validation loss.

Proposition 4 (Validation Loss Dataset Derivative). *Define \mathbf{L} as the loss function derivative with respect to the network outputs as,*

$$\mathbf{L} = \left[\frac{\partial \ell}{\partial f}(f(\mathbf{x}_1), y_1), \dots, \frac{\partial \ell}{\partial f}(f(\mathbf{x}_N), y_N) \right]$$

Then the dataset derivative importance weights with respect to final validation is given by

$$\nabla_\alpha L_{\text{val}}(\mathbf{w}_\alpha) = \mathbf{Z} \mathbf{C}_\alpha \mathbf{Z}^\top \times (\mathbf{L}^\top \mathbf{Y}^\top (\mathbf{I} - \mathbf{D}_\alpha \mathbf{Z} \mathbf{C}_\alpha \mathbf{Z}^\top)). \quad (2.10)$$



Figure 2.2: **Examples of the reweighting done by DIVA.** (Left) Samples from the FGVC Aircraft classification dataset that are up-weighted by DIVA and (Right) samples that are down-weighted because they increase the test error. Down-weighted samples tend to have planes in non-canonical poses, multiple planes, or not enough information to classify the plane correctly.

2.3.3 Leave-one-out optimization

It is common in AutoML to optimize the hyper-parameters with respect to a separate validation set. However, using a separate validation may not be practical in limited data settings, which are a main focus of dataset optimization. To remedy this, we now show that we can instead optimize α by minimizing a leave-one-out cross-validation loss that only requires a training set: where \mathbf{w}_α^{-i} are the optimal weights obtained by training with the loss (2.5) on the entire dataset \mathcal{D} *except* for the i -th sample (\mathbf{x}_i, y_i) . This may seem counter-intuitive, since we are optimizing the weights of the training samples using a validation loss defined on the training set itself. It is useful to recall that \mathbf{w}_α^{-i} minimizes the α -weighted L_2 loss on the training set (minus the i -th example):

$$\mathbf{w}_\alpha^{-i} = \arg \min_{\mathbf{w}} L_{\text{train}}^{-i}(\mathbf{w}, \mathcal{D}_\alpha) = \arg \min_{\mathbf{w}} \sum_{j \neq i} \alpha_j \|f_{\mathbf{w}}(\mathbf{x}_j) - y_j\|^2 + \lambda \|\mathbf{w}\|^2. \quad (2.11)$$

Meanwhile, α minimizes the *unweighted* validation loss (un-weighted version of (2.12)). This prevents the existence of degenerate solutions for α .

Computing L_{LOO} naively would require training n classifiers, but fortunately, in the case of a linear classifier with the L_2 loss, a more efficient closed-form solution exists [GS93, RL07].

Generalizing those results to the case of a weighted loss, we are able to derive the following expression.

Proposition 5. *Define*

$$\mathbf{R}_\alpha = \mathbf{Z}^\top \sqrt{\mathbf{D}_\alpha} (\mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Z} + \lambda \mathbf{I})^{-1} \sqrt{\mathbf{D}_\alpha} \mathbf{Z}$$

Then α -weighted LOOV predictions defined in (2.11) admit a closed-form solution:

$$f_{\mathbf{w}_\alpha^{-i}}(\mathbf{z}_i) = \left[\frac{\mathbf{R}_\alpha \sqrt{\mathbf{D}_\alpha} \mathbf{Y} - \text{diag}(\mathbf{R}_\alpha) \sqrt{\mathbf{D}_\alpha} \mathbf{Y}}{\text{diag}(\sqrt{\mathbf{D}_\alpha} - \sqrt{\mathbf{D}_\alpha} \mathbf{R}_\alpha)} \right]_i, \quad (2.12)$$

where $\text{diag}(\mathbf{A}) = [a_{11}, \dots, a_{nn}]$ denotes the vector containing the diagonal of \mathbf{A} , and the division between vectors is element-wise.

Note that the prediction $f_{\mathbf{w}_\alpha^{-i}}(\mathbf{z}_i)$ on the i -th sample when training on all the other samples is a differentiable function of α . Composing (2.12) into the un-weighted version of the loss on the training set, we compute the derivative $\nabla_\alpha L_{\text{LOO}}(\alpha)$, which allows us to optimize the cross-validation loss with respect to the sample weights, without the need of a separate validation set. We give the closed-form expression for $\nabla_\alpha L_{\text{LOO}}(\alpha)$ in Appendix 2.A.

2.3.4 Dataset optimization with DIVA

We can now apply the closed-form expressions for $\nabla_\alpha L_{\text{val}}(\alpha)$ and $\nabla_\alpha L_{\text{LOO}}(\alpha)$ for differentiable dataset optimization. We describe the optimization using L_{val} , but the same applies to L_{LOO} .

DIVA Reweight. The basic task consists in reweighting the samples of an existing dataset in order to improve generalization. This can curate a dataset by reducing the influence of outliers or wrong labels, or by reducing possible imbalances. To optimize the dataset weights, we use gradient descent in the form:

$$\alpha \leftarrow \alpha - \eta \nabla_\alpha L_{\text{val}}. \quad (2.13)$$

It is important to notice that L_{val} is an unbiased estimator of the test loss only at the first step, hence optimizing using (2.13) for multiple steps can lead to over-fitting (see Appendix

2.A). Therefore, we apply only 1-3 gradient optimization steps with a relatively large learning rate $\eta \simeq 0.1$. This early stopping both regularizes the solution and decreases the wall-clock time required by the method. We initialize α so that $\alpha_i = 1$ for all samples.

DIVA Extend. The dataset gradient also allows us to extend an existing dataset. Given a core dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ and an external (potentially noisy) data pool $\mathcal{E} = \{(\hat{\mathbf{x}}_i, \hat{y}_i)\}_{i=N+1}^{N+M}$, we want to find the best samples from \mathcal{E} to add to \mathcal{D} . For this we merge \mathcal{D} and \mathcal{E} in a single dataset and initialize α such that $\alpha_i = 1$ for samples of \mathcal{D} and $\alpha_i = 0$ for samples of \mathcal{E} (so that initially the weighted dataset matches \mathcal{D}). We then compute $\nabla_{\alpha} L_{\text{val}}(\alpha)$ to find the top k samples of \mathcal{E} that have the largest negative value of $\nabla_{\alpha} L_{\text{val}}(\alpha)_i$, i.e., the samples that would give the largest reduction in validation error if added to the training set and add them to \mathcal{D} . This is repeated until the remaining samples in \mathcal{E} all have positive value for the derivative (adding them would not further improve the performance).

Detrimental sample detection. The i -th component of $\nabla_{\alpha} L_{\text{val}}$ specifies the influence of the i -th sample on the validation loss. In particular, $(\nabla_{\alpha} L_{\text{val}})_i > 0$ implies that the sample increases the validation loss, hence it is detrimental (e.g., it is mislabeled or overly represented in the dataset). We can select the set of detrimental examples by thresholding $\nabla_{\alpha} L_{\text{val}}$:

$$\text{Detrimental}(\epsilon) = \{i : (\nabla_{\alpha} L_{\text{val}})_i \geq \epsilon\}. \quad (2.14)$$

2.4 Experimental results

For our models we use standard residual architectures (ResNet) models pre-trained on ImageNet [DDS09] and Places365 [ZLK17]. For our experiments on dataset optimization we consider datasets that are smaller than the large-scale datasets used for pre-training as we believe they reflect more realistic conditions for dataset optimization. For our experiments we use the CUB-200 [WBM10], FGVC-Aircraft, [MKR13], Stanford Cars [KSD13], Caltech-256 [GHP07], Oxford Flowers 102 [NZ08], MIT-67 Indoor [QT09], Street View House Number

Dataset	Original	DIVA Reweight	[CLM17]	[RZY18]	Gain
Aircrafts	57.58	54.64	70.48	81.82 (80.62)	+2.94
Cub-200	39.30	36.93	57.85	72.55 (75.35)	+2.36
MIT Indoor-67	32.54	31.27	37.84	64.48 (58.06)	+1.27
Oxford Flowers	20.23	19.16	22.82	48.80 (55.46)	+1.07
Stanford Cars	58.91	56.31	75.87	83.09 (84.50)	+2.56
Caltech-256	23.98	21.29	37.52	58.44 (52.77)	+2.69

Table 2.1: Test error of DIVA Reweight to curate several fine-grain classification datasets. We use a ResNet-34 pretrained on ImageNet as feature extractor and train a linear classifier on top of the last layer. Note that DIVA Reweight can improve performance even on curated and noiseless datasets whereas other reweighting methods based on hard-coded rules may be detrimental in this case.

[NWC11], and the Oxford Pets [PVZ12] visual recognition and classification datasets. In all experiments, we use the network as a fixed feature extractor, and train a linear classifier on top of the network features using the weighted L_2 loss (2.5) and optimize the α weights using DIVA.

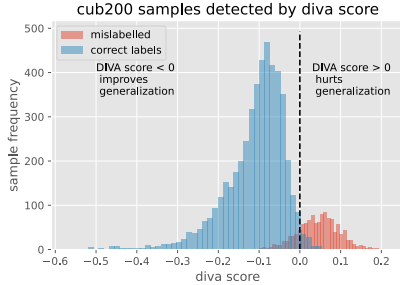
Dataset AutoCuration. We use DIVA Reweight to optimize the importance weights of samples from several fine-grain classification datasets. While the datasets have already been manually curated by experts to generally exclude out-of-distribution or mislabeled examples, we still observe that in all cases DIVA can further improve the test error of the model (Table 2.1). To understand how DIVA achieves this, in Figure 2.2 we show the most up-weighted (left) and down-weighted (right) examples on the FGVC Aircraft classification task [MKR13]. We observe that DIVA tends to give more weight to clear, canonical examples, while it detects as detrimental (and hence down-weights) examples that contain multiple planes (making the label uncertain), or that do not clearly show the plane, or show non-canonical

poses. We compare DIVA Reweight with two other re-weighting approaches: [RZY18], that applies re-weighting using information extracted from a separate validation gradient step, and [CLM17], which reweights based on the uncertainty of each prediction (threshold-closeness weighting scheme). For [RZY18], we set aside 20% of the training samples as validation for the reweight step, but use all samples for the final training (in parentheses). We notice that both baselines, which are designed to reweight noisy datasets, underperform with respect to DIVA on datasets without artificial noise.

Dataset extension. We test the capabilities of DIVA Extend to extend a dataset with additional samples of the distribution. In Figure 2.4 and Table 2.2, we observe that DIVA is able to select the most useful examples and reaches an optimal performance generalization error using significantly less samples than the baseline uniform selection. Moreover, we notice that DIVA identifies a smaller subset of samples that provides better test accuracy than adding all the pool samples to the training set.

Dataset	DIVA Extend	Uniform	Improvement
Aircrafts	58.00	60.01	+2.01
Cub-200	39.42	42.29	+2.87
MIT Indoor-67	32.54	33.73	+1.19
Oxford Flowers	20.56	23.29	+2.73
Stanford Cars	60.37	62.45	+2.09
Caltech-256	21.97	24.55	+2.59

Table 2.2: Results of using DIVA Extend to select the best samples to extend several fine-grain classification datasets. We train a linear classifier on top of a ResNet-34 pretrained on ImageNet, and compare the test performance when extending the target training dataset with 50% of the pool samples selected either uniformly at random or via DIVA Extend.



Dataset	F1-score ($\epsilon = 0$)	AUC
Cub200	0.87	0.98
Aircrafts	0.68	0.90
MIT Indoor-67	0.86	0.98
Stanford Cars	0.75	0.93
Caltech-256	0.92	0.99
Oxford Flowers	0.83	0.97

Figure 2.3: **(Left)** Distribution of LOO DIVA gradients for correctly labelled and mislabelled samples in CUB-200 dataset (20% of the samples are mislabeled by replacing their label uniformly at random). **(Right) DIVA for outlier rejection.** We use DIVA on a ResNet-34 network linearization and detect mislabelled samples (outliers) in a dataset present with 20% label noise. Selection is based on $\nabla_{\alpha}(L_{\text{val}}(\mathbf{w}_{\alpha}))_i > \epsilon$.

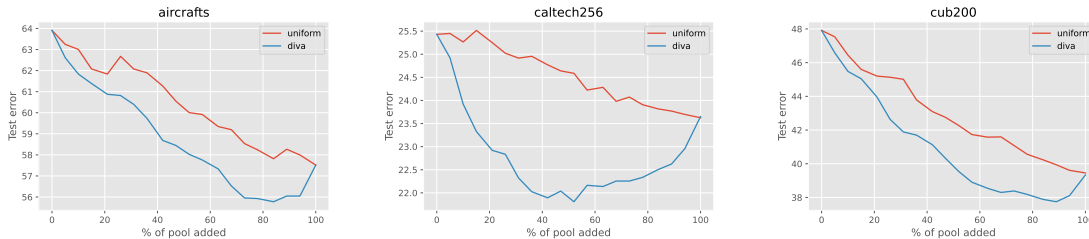


Figure 2.4: **DIVA Extend.** We show the test error achieved by the model as we extend a dataset with samples selected from a dataset pool using either DIVA Extend (red line) or uniform sampling (blue line). The pool set matches the same distribution as the training set. In all cases DIVA Extend outperforms uniform sampling and identifies subsets of the pool set with better performance than the whole pool. We also note that using only a subset selected by DIVA as opposed to using the whole pool, actually improves the test accuracy.

Detrimental sample detection. To test the ability of DIVA to detect training samples that are detrimental for generalization, we add a proportion of random labels in the dataset. Following the procedure outlined in Subsection 2.3.4 we detect outliers by thresholding where the derivative $\nabla_{\alpha} L_{\text{LOO}}(\alpha)_i$ is positive. In Figure 2.3 we plot the histogram of the derivatives for correct and mislabeled examples. We observe that majority of mislabeled examples have

a positive derivative while the vast majority of the correctly labelled samples have a negative derivative. In particular, we can directly classify an example as mislabeled if the derivative is positive. In Figure 2.3 we report the F1 score and AUC obtained in a mislabeled sample detection task using the DIVA gradients.

Multi-modal learning. Recent multi-modal models such as CLIP [RKH21] can embed both text and images in the same vector spaces. This allows to boost the performance on few-shot image classification tasks by also adding to the training set textual descriptions of the classes, such as the label name. However, training on label names may also hurt the performance, for example if the label name is not known by the CLIP model. To test this, we create a few-shot task by selecting 20 images per class from the Caltech-256. We then use DIVA Extend to select an increasing number of labels to add to the training set. In Figure 2.5 (right), we show that DIVA can select the beneficial label embeddings to add in order to improve the few-shot test performance. However, when forced to add all labels, including detrimental ones, the test error starts to increase again.

Data augmentation. To further test the versatility of DIVA, we qualitatively evaluate DIVA Reweight on the task of tuning the weights with which we apply a given data augmentation procedure. Let t_1, \dots, t_K be a set of data augmentation transformations. Let \mathcal{D}^{t_k} be the result of applying the data augmentation t_k to \mathcal{D} . We can create an augmented dataset $\mathcal{D}^{\text{aug}} = \mathcal{D} \cup \mathcal{D}^{t_0} \cup \dots \cup \mathcal{D}^{t_K}$, by merging all transformed datasets. We then apply DIVA Reweight on \mathcal{D}^{aug} to optimize the weight α of the samples. Based on the updated importance weights we estimate the optimal probability with which to apply the transformation t_k as $p_k = (\sum_{i \in \mathcal{D}^{t_k}} \alpha_i) / (\sum_i \alpha_i)$. In particular we select common data augmentation procedures, horizontal flip and vertical flip, and we tune their weights on the Street View House Number, Oxford Flowers and the Oxford Pets classification tasks. We observe that DIVA assigns different weights to each transformation aligning with the task (Figure 2.5): on the number classification task DIVA penalizes both vertical and horizontal flips, which may confuse

different classes (such 2 and 5, 6 and 9). On an animal classification task (Oxford Pets) DIVA does not penalize horizontal flips, but penalizes vertical flips since they are out of distributions. Finally, on Flowers classification DIVA gives equal weights to all transformations (most flower pictures are frontal so all rotations and flips are valid).

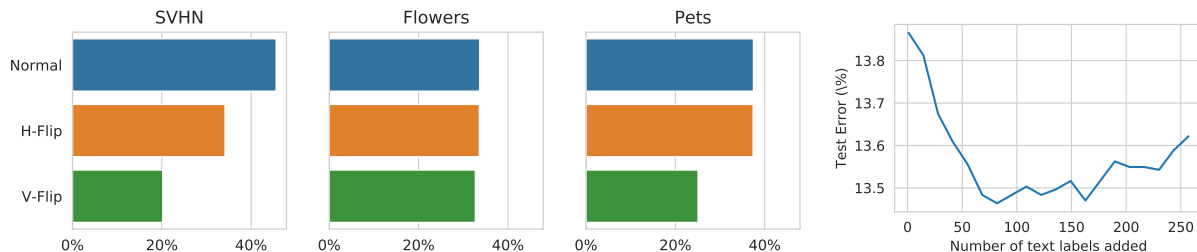


Figure 2.5: **(Left) DIVA can select the best data augmentation for each task.** We optimize the weight with which each data augment transformation is applied. DIVA optimizes the transformation to apply for the particular task. **(Right) Use of DIVA extend on a multi-modal task.** Selecting only the most beneficial text embeddings to use in a multi-modal classification task (as scored by DIVA) outperforms blindly using all available text embeddings.

2.5 Discussion

In this chapter we present a gradient-based method to optimize a dataset. In particular we focus on sample reweighting, extending datasets, and removing outliers from noisy datasets. We note that by developing the notion of a dataset derivative we are capable of improving dataset quality in multiple disparate problems in machine learning. The dataset derivative we present is given in closed-form and enables general reweighting operations on datasets based on desired differentiable validation losses. In cases where a set-aside validation loss is not available we show the use of the leave-one-out framework enables computing and optimizing a dataset “for free” and derive the first closed-form dataset derivative based on the LOO framework.

2.A Appendix

We structure the appendix as follows: We present additional experiments in Subsection 2.A.1 and we describe the details of the experiments from Section 2.4 in Subsection 2.A.2. In Subsection 2.A.3 we provide proofs for the propositions of the chapter and additional discussion on the methods.

2.A.1 Additional experiments

Validation overfitting. When updating the dataset using the dataset derivative there is a risk of overfitting to the validation set after repeated applications of the derivative. Namely the validation loss is initially an unbiased estimate of the test loss yet after using it to update the dataset repeatedly it eventually will start overfitting. In our settings, we notice that when using a small number of gradient updates (< 5) and with step sizes $\eta \sim 0.1$ we are able to avoid overfitting and improve the *test* error when optimizing the dataset derivative based on the validation loss. In this experiment we present the final test and validation classification errors of optimized datasets. As we optimize with respect to the validation loss, it is indeed clear that the validation loss decreases dramatically yet more importantly are the effect of the test accuracy.

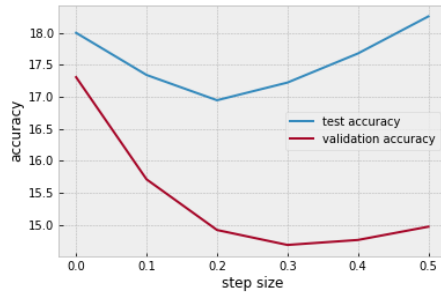


Figure 2.6: The test error decreases as the dataset is optimized with respect to the validation set until eventually overfitting commences. The validation set error decreases more significantly as the dataset is optimized directly on the validation set, yet for very large step-sizes the first order optimization becomes inaccurate. The plot uses the Caltech-256 dataset to illustrate the overfitting

DIVA Extend plots. In Figure 2.4 we report the results on all the remaining datasets following the set-up of Figure 2.4. In Table 2.2 we report the accuracy of DIVA Extend and uniform sampling on the various datasets when adding 50% of samples from the pool.

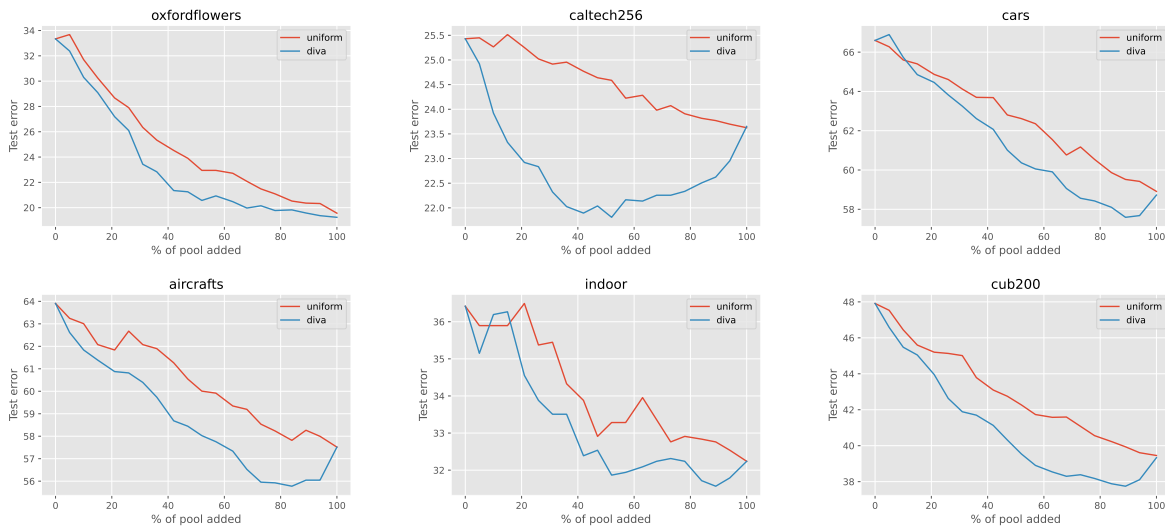


Figure 2.7: Same plots as Figure 2.4 on the other fine-grained datasets.

2.A.2 Experimental details

Dataset details For our experiments we utilize several fine-grain classification datasets from the computer vision community that are standard for fine-tuning image classification tasks (CUB-200 [WBM10], FGVC-Aircraft, [MKR13], Stanford Cars [KSD13], Caltech-256 [GHP07], Oxford Flowers 102 [NZ08], MIT-67 Indoor [QT09], Street View House Number [NWC11], and the Oxford Pets [PVZ12]). Some of the datasets do not follow a default train-test split and we use the following splits commonly used in the literature for the datasets,

- Oxford Flowers 102: We use the original 1020 images in the training split without incorporating the default validation set.
- Caltech-256: We split the dataset into a training set with 60 images from each class for training, and use remaining data for testing.

Pre-training setup For the pre-trained networks we use for fine-tuning, we use the pre-trained configurations available on PyTorch’s torchvision package. In particular the reference networks are pre-trained using the ImageNet [DDS09] dataset. The images embedded by the network are pre-processed via standard resizing and center cropping (256 resize, followed by a 224 cropping).

Regularization parameter λ : To get the best unweighted dataset baseline to compare with the optimized dataset, for each of the un-optimized original datasets, we first search for optimal λ values in $\lambda \in \{2^n \text{ for } n \in \{-20, -19, \dots, 4\}\}$ to measure the classifier’s performance. After selecting optimal λ we proceed with dataset optimization with the optimal λ values. Note that DIVA does not require λ to be optimal and improvements are even more significant for un-optimized λ .

Dataset Derivative Computation In Section 2.A.3 we derive the closed-form dataset derivatives used for DIVA. We computed the closed-form solution analytically and we verified our results using automatic differentiation tools on large number of conditions including synthetic and real data, as an additional method to verify the correctness of the derivative formulas.

DIVA method details

DIVA Extend For Table 2.2 and Figure 2.4 we first split the original training set into 50% training subset and 50% pool subset that will be used to selectively extend the training subset with DIVA or other extension approaches. We run DIVA Extend LOO and uniform sampling to add pool samples to the training set. In both settings we incrementally extend the training subset from the pool (For DIVA, by selecting samples with top DIVA score) in each step we extend an equal number of samples $((\# \text{ pool samples}) // (\# \text{ number of steps}))$.

In the figure we present the test error as a function of training set size and compare DIVA sample selection with selecting the same number of samples at uniform from the pool set. In Table 2.2, we present the improvement in test accuracy at the 50% extend mark of the pool set (e.g. extending 25% of the original training set) between DIVA and uniform sample selection of the same number of samples. For both experiments we use the ResNet-34 architecture.

DIVA Reweigh In Table 2.1 we use DIVA Reweight LOO with the same parameters for all of the datasets we consider. The DIVA parameters we use are $K = 4$ for the number of steps and $\eta = 0.15$ for the step-size. As with DIVA Extend, we use ResNet 34 for the architecture for the representation.

DIVA validation loss We find the cross entropy loss to work better as the loss function applied to the validation predictions (both in LOO and regular validation). Further for

LOO we find it crucial to apply the validation loss only on mis-classified LOO predictions to improve the test accuracy of the model, this can be interpreted as the cross entropy loss with a “hard margin hinge” loss.

Outlier rejection For the results presented in the side-by-side Table and Figure 2.3 we apply 20% random label noise to each class in the dataset and use DIVA LOO to compute the normalized DIVA gradient (DIVA score) of each sample. The F1 score reports classification by thresholding with $\epsilon = 0$ and the AUC is computing by thresholds spanning detection of no samples, to detection of all samples.

2.A.3 Proofs of propositions

We write the proposition statements for convenience.

Proposition (Model-Dataset Derivative $\nabla_{\alpha} \mathbf{w}_{\alpha}$). *For the ridge regression problem Equation (2.5) and \mathbf{w}_{α} defined as in Equation (2.6), define*

$$\mathbf{C}_{\alpha} = (\mathbf{Z}^{\top} \mathbf{D}_{\alpha} \mathbf{Z} + \lambda \mathbf{I})^{-1}. \quad (2.15)$$

Then the Jacobian of \mathbf{w}_{α} with respect to α is given by

$$\nabla_{\alpha} \mathbf{w}_{\alpha} = \mathbf{Z} \mathbf{C}_{\alpha} \circ ((\mathbf{I} - \mathbf{Z} \mathbf{C}_{\alpha} \mathbf{Z}^{\top} \mathbf{D}_{\alpha}) \mathbf{Y}). \quad (2.16)$$

Proof of Proposition 3:

We recall the settings of the problem are $\mathbf{Z} \in \mathbb{R}^{n \times m}$, $\mathbf{Y} \in \mathbb{R}^{n \times k}$, $\mathbf{w}_{\alpha} \in \mathbb{R}^{m \times k}$ and the importance weights $\alpha \in \mathbb{R}^n$. Here n is the number of samples, m is the number of parameters for each output of the model, and k is the number of classes (represented via the one-hot convention).

The derivation of $\nabla_{\alpha} \mathbf{w}_{\alpha}$ we present is computational in nature and without the loss of generality we consider the derivative for a single output class $k = 1$ (one-vs-all classification naturally extends). For the single class settings, the relevant dimensions are $\mathbf{w}_{\alpha} \in \mathbb{R}^m$ and $\nabla_{\alpha} \mathbf{w}_{\alpha} \in \mathbb{R}^{m \times n}$ (for numerator layout convention of the derivative). To further simplify we

consider the derivative entrywise for single index α_r ,

$$\frac{\partial \mathbf{w}_\alpha}{\partial \alpha_r} \in \mathbb{R}^{m \times 1}.$$

The closed-form solution is,

$$\mathbf{w}_\alpha = \mathbf{C}_\alpha \mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Y}.$$

using chain rule,

$$\frac{\partial \mathbf{w}_\alpha}{\partial \alpha_r} = \frac{\partial \mathbf{C}_\alpha}{\partial \alpha_r} \times \mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Y} + \mathbf{C}_\alpha \times \frac{\partial \mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Y}}{\partial \alpha_r}.$$

We compute the two parts of the derivative in turn:

Let $\mathbf{K}_\alpha = (\mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Z} + \lambda \mathbf{I})$ so that $\mathbf{K}_\alpha = \mathbf{C}_\alpha^{-1}$. Then

$$\frac{\partial \mathbf{K}_\alpha}{\partial \alpha_r} = \frac{\partial}{\partial \alpha_r} (\mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Z} + \lambda \mathbf{I}) = \mathbf{Z}_{r,:}^\top \mathbf{Z}_{r,:}$$

where $\mathbf{Z}_{r,:} \in \mathbb{R}^{1 \times m}$ is the r th row of \mathbf{Z} . Next note,

$$\mathbf{C}_\alpha \mathbf{K}_\alpha = \mathbf{I}$$

differentiating both sides,

$$\frac{\partial \mathbf{C}_\alpha \mathbf{K}_\alpha}{\partial \alpha_r} = \mathbf{0}.$$

Applying chain rule we get,

$$\frac{\partial \mathbf{C}_\alpha \mathbf{K}_\alpha}{\partial \alpha_r} = \frac{\partial \mathbf{C}_\alpha}{\partial \alpha_r} \mathbf{K}_\alpha + \mathbf{C}_\alpha \frac{\partial \mathbf{K}_\alpha}{\partial \alpha_r},$$

rearranging, substituting $\frac{\partial \mathbf{K}_\alpha}{\partial \alpha_r}$, and multiplying to the right by \mathbf{C}_α

$$\frac{\partial \mathbf{C}_\alpha}{\partial \alpha_r} = -\mathbf{C}_\alpha \mathbf{Z}_{r,:}^\top \mathbf{Z}_{r,:} \mathbf{C}_\alpha.$$

Next, by direct computation $\frac{\partial \mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Y}}{\partial \alpha_r}$ satisfies

$$\frac{\partial \mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Y}}{\partial \alpha_r} = y_r \cdot \mathbf{Z}_{r,:}^\top.$$

Combining the original terms we have

$$\begin{aligned} \frac{\partial \mathbf{w}_\alpha}{\partial \alpha_r} &= -(\mathbf{C}_\alpha \mathbf{Z}_{r,:}^\top \mathbf{Z}_{r,:} \mathbf{C}_\alpha) \times \mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Y} + y_r \cdot \mathbf{C}_\alpha \times \mathbf{Z}_{r,:}^\top \\ &= \mathbf{C}_\alpha \mathbf{Z}_{r,:}^\top (y_r - \mathbf{Z}_{r,:} \mathbf{C}_\alpha \mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Y}). \end{aligned}$$

Now the full derivative is written as

$$\nabla_{\alpha} \mathbf{w}_{\alpha} = \mathbf{ZC}_{\alpha} \circ \left((\mathbf{I} - \mathbf{ZC}_{\alpha} \mathbf{Z}^{\top} \mathbf{D}_{\alpha}) \mathbf{Y} \right)$$

□

Proposition (Validation Loss Dataset Derivative). *Define \mathbf{L} as the matrix of the loss function derivative with respect to network training outputs as,*

$$\mathbf{L} = \left[\frac{\partial \ell}{\partial f}(f_{\mathbf{w}_{\alpha}}(\mathbf{x}_1), y_1), \dots, \frac{\partial \ell}{\partial f}(f(\mathbf{x}_N), y_N) \right].$$

Then the dataset derivative of the importance weights with respect to final validation loss is given by

$$\nabla_{\alpha} L_{\text{val}}(\mathbf{w}_{\alpha}) = \mathbf{ZC}_{\alpha} \mathbf{Z}^{\top} \times (\mathbf{L}^{\top} \mathbf{Y}^{\top} (\mathbf{I} - \mathbf{D}_{\alpha} \mathbf{ZC}_{\alpha} \mathbf{Z}^{\top})). \quad (2.17)$$

Proof of Proposition 4:

This follows from the chain rule combined with simplification of broadcasting terms. We again consider the single output settings with the single coordinate derivative $\frac{\partial L_{\text{val}}}{\partial \alpha_r}$ which is given as,

$$\frac{\partial L_{\text{val}}}{\partial \alpha_r} = \nabla_{\mathbf{w}} L_{\text{val}} \frac{\partial \mathbf{w}_{\alpha}}{\partial \alpha_r}.$$

With

$$\begin{aligned} &= \sum_{i=1}^n \mathbf{L}_{:,i} \times \mathbf{z}_i^{\top} \\ &= \mathbf{L}^{\top} \mathbf{Z}. \end{aligned}$$

Therefore

$$\left(\nabla_{\alpha} L_{\text{val}}(\mathbf{w}_{\alpha}) \right)_i = \sum_{j,k} \left(\nabla_{\alpha} \mathbf{w}_{\alpha} \right)_{i,j,k} (\mathbf{L}^{\top} \mathbf{Z})_{j,k}.$$

Now \mathbf{L}, \mathbf{Z} can be separated into the two terms of $\nabla_{\alpha} \mathbf{w}_{\alpha}$,

$$\nabla_{\alpha} L_{\text{val}}(\mathbf{w}_{\alpha}) = \mathbf{ZC}_{\alpha} \mathbf{Z}^{\top} \times (\mathbf{L}^{\top} \mathbf{Y}^{\top} (\mathbf{I} - \mathbf{D}_{\alpha} \mathbf{ZC}_{\alpha} \mathbf{Z}^{\top})) \quad (2.18)$$

□

Next we consider the derivations for the Leave One Out (LOO) framework. In the LOO framework one applies cross-validation to a training set $\{(\mathbf{z}_1, y_1), \dots, (\mathbf{z}_n, y_n)\}$ by running n -fold cross validation, where in each fold, the i th sample (\mathbf{z}_i, y_i) is taken out and is used for validation while the optimal classifier is solved for the remaining of the training task,

$$\mathbf{w}_{-i} = \arg \min_{\mathbf{w}} \sum_{j \neq i} \ell(f(\mathbf{z}_j), y_j). \quad (2.19)$$

Then the LOO prediction at the i th index is defined as $(\mathbf{f}_{LOO})_i = f_{\mathbf{w}_{-i}}(\mathbf{z}_i)$. Below we prove the LOO predictions can be written in closed-form without explicit cross validation calculations.

Proposition 6 (Closed-form LOO prediction vector). *Define the LOO vector predictions as,*

$$\mathbf{f}_{LOO} = [f_{\mathbf{w}_{-1}}(\mathbf{z}_1), \dots, f_{\mathbf{w}_{-n}}(\mathbf{z}_n)]^\top$$

and define

$$\mathbf{R} = \mathbf{Z}^\top (\mathbf{Z}^\top \mathbf{Z} + \lambda \mathbf{I})^{-1} \mathbf{Z}$$

then for the learning task Equation (2.19) LOO predictions are given as

$$\mathbf{f}_{LOO} = \frac{\mathbf{R}\mathbf{y} - \text{diag}(\mathbf{R})\mathbf{y}}{\mathbf{I} - \text{diag}(\mathbf{R})}. \quad (2.20)$$

Proof of Proposition 6:

The proof is reproduced from [RL07] for completeness.

Without the loss of generality we derive the LOO prediction of \mathbf{z}_n . Namely given, $\{(\mathbf{z}_1, y_1), \dots, (\mathbf{z}_n, y_n)\}$ we use $\{(\mathbf{z}_1, y_1), \dots, (\mathbf{z}_{n-1}, y_{n-1})\}$ for training and validate using $\{(\mathbf{z}_n, y_n)\}$. Denote \mathbf{w}^{-n} as be the optimal solution to this training task with regularization parameter λ and define the (currently unknown) LOO prediction as

$$q = f_{\mathbf{w}^{-n}}(\mathbf{z}_n).$$

We define the modified learning task consisting of $\{(\mathbf{z}_1, y_1), \dots, (\mathbf{z}_{n-1}, y_{n-1}), (\mathbf{z}_n, q)\}$ where we added the data point (\mathbf{z}_n, q) . Note that the optimal solution with λ regularization to the

modified learning task is again \mathbf{w}^{-n} since q is taken to have a zero residual. therefore the solution to the modified learning problem can be written in closed-form as,

$$\mathbf{w}^{-n} = (\mathbf{Z}^\top \mathbf{Z} + \lambda \mathbf{I})^{-1} \mathbf{Z}^\top [\mathbf{y}_{1:n-1}, q]^\top.$$

Using \mathbf{w}^{-n} we write the equation for the LOOV prediction q ,

$$q = \langle \mathbf{z}_n, \mathbf{w}^{-i} \rangle = \mathbf{z}_n^\top (\mathbf{Z}^\top \mathbf{Z} + \lambda \mathbf{I})^{-1} \mathbf{Z}^\top [\mathbf{y}_{1:n-1}, q]^\top.$$

Let $\mathbf{R} = \mathbf{Z}(\mathbf{Z}^\top \mathbf{Z} + \lambda \mathbf{I})^{-1} \mathbf{Z}^\top$ then we have

$$q = \mathbf{z}_n^\top \mathbf{w}_{-i} = \mathbf{R}_{n,:} [\mathbf{Y}_{1:n-1}, q]^\top$$

and by adding and subtracting $\mathbf{R}_{:,n}$ multiplied by $[0, y_n]^\top$ we get,

$$q = \mathbf{z}_n^\top \mathbf{w}_{-i} = \mathbf{R}_{:n-1,n} [\mathbf{Y}_{1:n-1}]^\top + \mathbf{R}_{:,n} [0, q] + \mathbf{R}_{:,n} [0, y_n]^\top - \mathbf{R}_{:,n} [0, y_n]^\top.$$

Re-arranging we have

$$q - \mathbf{R}_{nn} q = \mathbf{R}_{:,n} \mathbf{y} - y_n \mathbf{R}_{n,n}$$

Solving for q we get,

$$q = \frac{\mathbf{R}_{:,n} \mathbf{y} - y_n \mathbf{R}_{n,n}}{1 - \mathbf{R}_{n,n}}$$

And without the loss of generality the full prediction vector is given as,

$$\boxed{\mathbf{f}_{\text{LOO}} = \frac{\mathbf{R} \mathbf{y} - \text{diag}(\mathbf{R}) \mathbf{y}}{\text{diag}(\mathbf{I} - \mathbf{R})}} \quad (2.21)$$

□

□

Given the closed-form LOOV expression we may use \mathbf{f}_{LOO} for the validation loss to compute the dataset derivative on L_{val} without any additional validation data. While this may seem contradictory as we are optimizing the dataset validated via the weighting duality between sample loss weighting and data scaling, we define the leave one out value predictions in the weighted dataset settings and evaluate on the original (unweighted) data points as,

$$f_{\mathbf{w}_\alpha^{-i}}(\mathbf{z}_i) \quad (2.22)$$

with,

$$\mathbf{w}_\alpha^{-i} = \arg \min_{\mathbf{w}} \sum_{j \neq i} \alpha_j \ell(f(\mathbf{z}_j), y_j). \quad (2.23)$$

Therefore the α -weighted LOO term is $f_{\mathbf{w}_\alpha^{-i}}(\mathbf{z}_i)$ is faithful to the original distribution despite being trained with the weighted loss Equation (2.23). We in fact are able to show that α -weighted LOO formulation also admits a closed-form solution that satisfies our definition and for DIVA LOO we utilize the derivative of the closed-form to optimize the dataset.

Proposition. *Define*

$$\mathbf{R}_\alpha = \mathbf{Z}^\top \sqrt{\mathbf{D}_\alpha} (\mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Z} + \lambda \mathbf{I})^{-1} \sqrt{\mathbf{D}_\alpha} \mathbf{Z}$$

Then the α -weighted LOOV predictions defined in Equation (2.11) admit a closed-form solution:

$$f_{\mathbf{w}_\alpha^{-i}}(\mathbf{z}_i) = \left[\frac{\mathbf{R}_\alpha \sqrt{\mathbf{D}_\alpha} \mathbf{Y} - \text{diag}(\mathbf{R}_\alpha) \sqrt{\mathbf{D}_\alpha} \mathbf{Y}}{\text{diag}(\sqrt{\mathbf{D}_\alpha} - \sqrt{\mathbf{D}_\alpha} \mathbf{R}_\alpha)} \right]_i, \quad (2.24)$$

Further the LOO dataset derivative is well-defined and satisfies the following gradient condition,

$$\text{diag}(\nabla_\alpha \mathbf{f}_{LOO}) = \mathbf{0}. \quad (2.25)$$

The gradient condition $\text{diag}(\nabla_\alpha \mathbf{f}_{LOO}) = \mathbf{0}$ implies that the LOO prediction at \mathbf{z}_i does not depend on α_i and ensures that the closed-form solution is well defined in the α -weighted settings and is differentiable.

Proof of Proposition 5:

The proof builds on Proposition 6 for the weighted settings.

In general since LOO expression describes the weighting problem, it must be shown that the introduction of the weights do not break the argument of Proposition 6. Considering the optimization problem in Equation (2.11). We also use the duality between the loss weights and data scaling to note that \mathbf{w}_α^{-i} can be derived by considering the unweighted LOO Equation (2.19) with the modified dataset,

$$\{(\sqrt{\alpha_1} \mathbf{z}_1, \sqrt{\alpha_1} y_1), \dots, (\sqrt{\alpha_n} \mathbf{z}_n, \sqrt{\alpha_n} y_n)\}. \quad (2.26)$$

Indeed in this settings with the newly defined data the derivation in Proposition 6 of the final prediction vector of the LOO entries holds with the same optimal solution \mathbf{w}_α due to the duality between data scaling and loss weights. Nonetheless the closed-form LOO predictions \mathbf{f}_{LOO} are evaluated at the data-points $\sqrt{\alpha_i}\mathbf{z}_i$. Since the model is linear the final predictions at the original data-points of the weighted training settings are written as $\mathbf{f}_{\text{LOO}}/\sqrt{\alpha}$.

Noting the weighted training problem can be expressed as $\tilde{\mathbf{Y}} = \sqrt{\mathbf{D}_\alpha}\mathbf{Y}$, $\tilde{\mathbf{Z}} = \sqrt{\mathbf{D}_\alpha}\mathbf{Z}$ we use Proposition 6 to write analogously

$$\mathbf{R}_\alpha = \mathbf{Z}^\top \sqrt{\mathbf{D}_\alpha} (\mathbf{Z}^\top \mathbf{D}_\alpha \mathbf{Z} + \lambda \mathbf{I})^{-1} \sqrt{\mathbf{D}_\alpha} \mathbf{Z} \quad (2.27)$$

and divide by $\sqrt{\alpha}$ by multiplying the denominator by $\sqrt{\mathbf{D}_\alpha}$,

$$f_{\mathbf{w}_\alpha^{-i}}(\mathbf{z}_i) = \left[\frac{\mathbf{R}_\alpha \sqrt{\mathbf{D}_\alpha} \mathbf{Y} - \text{diag}(\mathbf{R}_\alpha) \sqrt{\mathbf{D}_\alpha} \mathbf{Y}}{\text{diag}(\sqrt{\mathbf{D}_\alpha} - \sqrt{\mathbf{D}_\alpha} \mathbf{R}_\alpha)} \right]_i, \quad (2.28)$$

Since the derivation at the i th index of the weighted LOO prediction, $(\mathbf{f}_{\text{LOO}})_i = (\mathbf{w}_\alpha^{-i})^\top \mathbf{z}_i$ is entirely independent of α_i , we have

$$\frac{\partial (f_{\mathbf{w}_\alpha^{-i}}(\mathbf{z}_i))_i}{\partial \alpha_i} = 0.$$

□

CHAPTER 3

On the dynamics and convergence of Weight Normalization for training neural networks*

3.1 Introduction

In this chapter we prove the first result showing that the non-convex problem of optimizing a ReLU neural network that is trained with normalization layers converges to a global minima. The guarantee for the convergence of a global minima is counter-intuitive given the non-convexity of the problem that is exacerbated by applying normalization layers. However, using careful concentration of measure it is possible to illustrate descent of the objective in a linear rate. Dynamic normalization in the training of neural networks amounts to the application of an intermediate normalization procedure between layers of the network. Such methods have become ubiquitous in the training of neural nets since in practice they significantly improve the convergence speed and stability. This type of approach was popularized with the introduction of Batch Normalization (BN) [IS15] which implements a dynamic re-parametrization, normalizing the first two moments of the outputs at each layer over mini-batches. A plethora of additional normalization methods followed BN, notably including Layer Normalization (LN) [BKH16] and Weight Normalization (WN) [SK16]. Despite the impressive empirical results and massive popularity of dynamic normalization methods, explaining their utility and proving that they converge when training with non-smooth, non-convex loss functions has remained an unsolved problem. In this chapter

*This chapter is adapted from [DGM20]

we provide sufficient conditions on the data, initialization, and over-parametrization for dynamically normalized ReLU networks to converge to a global minimum of the loss function. For the theory we present we focus on WN, which is a widely used normalization layer in training of neural networks. WN was proposed as a method that emulates BN. It normalizes the input weight vector of each unit and separates the scale into an independent parameter. The WN re-parametrization is very similar to BN (see Section 3.3) and benefits from similar stability and convergence properties. Moreover, WN has the advantage of not requiring a batch setting, therefore considerably reducing the computational overhead that is imposed by BN [GG17].

When introducing normalization methods, the function parametrization defined by the network becomes scale invariant in the sense that re-scaling of the weights does not change the represented function. This re-scaling invariance changes the geometry of the optimization landscape drastically. To better understand this we analyze weight normalization in a given layer.

We consider the class of 2-layer ReLU neural networks which represent functions $f: \mathbb{R}^d \rightarrow \mathbb{R}$ parameterized by $(\mathbf{W}, \mathbf{c}) \in \mathbb{R}^{m \times d} \times \mathbb{R}^m$ as

$$f(\mathbf{x}; \mathbf{W}, \mathbf{c}) = \frac{1}{\sqrt{m}} \sum_{k=1}^m c_k \sigma(\mathbf{w}_k^\top \mathbf{x}). \quad (3.1)$$

Here we use the ReLU activation function $\sigma(s) = \max\{s, 0\}$ [NH10], m denotes the width of the hidden layer, and the output is normalized accordingly by a factor \sqrt{m} . We investigate gradient descent training with WN for (3.1), which re-parametrizes the functions in terms of $(\mathbf{V}, \mathbf{g}, \mathbf{c}) \in \mathbb{R}^{m \times d} \times \mathbb{R}^m \times \mathbb{R}^m$ as

$$f(\mathbf{x}; \mathbf{V}, \mathbf{g}, \mathbf{c}) = \frac{1}{\sqrt{m}} \sum_{k=1}^m c_k \sigma\left(g_k \cdot \frac{\mathbf{v}_k^\top \mathbf{x}}{\|\mathbf{v}_k\|_2}\right). \quad (3.2)$$

This gives a similar parametrization to [DLT18] that study convergence of gradient optimization of convolutional filters on Gaussian data. We consider a regression task, the L^2 loss, a random parameter initialization, and focus on the over-parametrized regime, meaning that $m > n$, where n is the number of training samples. Further, we make little to no assumptions about the data.

The focus this chapter is in analyzing neural network optimization with weight normalization layers. We rigorously derive the dynamics of weight normalization training and its convergence from the perspective of the Neural Tangent Kernel (NTK). Compared with un-normalized training, we prove that normalized networks follow a modified kernel evolution that features a “length-direction” decomposition of the NTK. This leads to two convergence regimes in WN training and explains the utility of WN from the perspective of the NTK. In the settings considered, WN significantly reduces the amount of over-parametrization needed for provable convergence, as compared with un-normalized settings. Further, we present a more careful analysis that leads to improved over-parametrization bounds as compared with [DZP19].

The main contributions of the work in this chapter are:

- We prove the first general convergence result for 2-layer ReLU networks trained with a normalization layer and gradient descent. Our formulation does not assume the existence of a teacher network and has only very mild assumptions on the training data.
- We hypothesize the utility of normalization methods via a decomposition of the neural tangent kernel. In the analysis we highlight two distinct convergence regimes and show how Weight Normalization can be related to natural gradients and enable faster convergence.
- We show that finite-step gradient descent converges for all weight magnitudes at initialization. Further, we significantly reduce the amount of over-parametrization required for provable convergence as compared with un-normalized training.

The rest of the chapter is organized as follows. We discuss related work in Section 3.2. In Section 3.3 we provide background on WN and derive key evolution dynamics of training in Section 3.4. We present and discuss our main results, alongside with the idea of the proof, in Section 3.5. Lastly we offer a discussion of our results and analysis in Section 3.6. Proofs are presented in the chapter appendices.

3.2 Related work

The neural network function class (3.1) has been studied in many papers including [ADH19, DZP19, ZMG19, WDW19] along with other similar over-parameterized architectures [ALL19a, LL18, DLT18]. An exuberant series of recent works prove that feed-forward ReLU networks converge to zero training error when trained with gradient descent from random initialization. Nonetheless, to the best of our knowledge, there are no proofs that ReLU networks trained with *normalization* on general data converge to a global minimum. This is in part because normalization methods completely change the optimization landscape during training. Here we show that neural networks of the form given above (3.2) converge at a linear rate when trained with gradient descent and WN. The analysis is based on the over-parametrization of the networks, which allows for guaranteed descent while the gradient is non-zero.

For regression training, a group of papers studied the trajectory of the networks’ predictions and showed that they evolve via a “neural tangent kernel” (NTK) as introduced by [JGH18]. The latter paper studies neural network convergence in the continuous limit of infinite width over-parametrization, while the works of [DZP19, ADH19, WDW19, ZMG19, OS19] analyze the finite width setting. For finite-width over-parameterized networks, the training evolution also exhibits a kernel that takes the form of a Gram matrix. In these works, the convergence rate is dictated by the least eigenvalue of the kernel. We build on this fact, and also on the general ideas of the proof of [DZP19] and the refined work of [ADH19].

Normalization methods theory A number of recent works attempt to explain the dynamics and utility of various normalization methods in deep learning. The original works on BN [IS15] and WN [SK16] suggest that normalization procedures improve training by fixing the intermediate layers’ output distributions. The works of [BGS18] and [STI18] argue that BN may improve optimization by improving smoothness of the Hessian of the loss, therefore allowing for larger step-sizes with reduced instability. [HHS17] showed that the effective step-size in BN is divided by the magnitude of the weights. This followed

the work on WNgrad [WWB18] that introduces an adaptive step-size algorithm based on this fact. Following the intuition of WNGrad, [ALL19b] proved that for smooth loss and network functions, the diminishing “effective step-size” of normalization methods leads to convergence with optimal convergence rate for properly initialized step-sizes. The work of [KDL19] explains the accelerated convergence of BN from a “length-direction decoupling” perspective. The authors along with [CLS19] analyze the linear least squares regime, with [KDL19] presenting a bisection method for finding the optimal weights. Robustness and regularization of Batch Normalization is investigated by [LWS18] and improved generalization is analyzed empirically. Shortly after the original work of WN, [YKO17] showed that for a single precptron WN may speed-up training and emphasized the importance of the norm of the initial weights. Additional stability properties were studied by [YPR19] via mean-field analysis. The authors show that gradient instability is inevitable even with BN as the number of layers increases; this is in agreement with [BFL17] for networks with residual connections. The work of [ACB19] suggests initialization strategies for WN and derives lower bounds on the width to guarantee same order gradients across the layers.

Over-parametrized neural networks There has been a significant amount of recent literature studying the convergence of un-normalized over-parametrized neural networks. In the majority of these works the analysis relies on the width of the layers. These include 2-layer networks trained with Gaussian inputs and outputs from a teacher network [Tia17, LY17] and [DLT18] (with WN). Assumptions on the data distribution are relaxed in [DZP19] and the works that followed [ZMG19, ADH19, WDW19]. Our proof technique follows the mechanism presented in this chain of works. [WDW19] extend convergence results to adaptive step-size methods and propose AdaLoss. Recently, the global convergence of over-parameterized neural networks was also extended to deep architectures [DLL19a, ALS19, ZCZ20, ZG19]. In the context of the NTK, [ZMG19] have proved fast convergence of neural networks trained with natural gradient methods and the K-FAC approximation [MG15]. In the over-parameterized regimes, [ADH19] develop generalization properties for the networks of the form (3.1). In

addition, in the context of generalization, [ALL19a] illustrates good generalization for deep neural networks trained with gradient descent. [CG20] and [CG19] derive generalization error bounds of gradient descent and stochastic gradient descent for learning over-parametrization deep ReLU neural networks.

3.3 Weight Normalization

Here we give an overview of the WN procedure and review some known properties of normalization methods.

Notation We follow the general convention of the thesis: lowercase, lowercase boldface, and uppercase boldface letters denote scalars, vectors and matrices respectively. We denote the Rademacher distribution as $U\{1, -1\}$ and write $N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for a Gaussian with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. Training points are denoted by $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$ and parameters of the first layer by $\mathbf{v}_k \in \mathbb{R}^d, k = 1, \dots, m$. We use $\sigma(x) := \max\{x, 0\}$, and write $\|\cdot\|_2, \|\cdot\|_F$ for the spectral and Frobenius norms for matrices. $\lambda_{\min}(\mathbf{A})$ is used to denote the minimum eigenvalue of a matrix \mathbf{A} and $\langle \cdot, \cdot \rangle$ denotes the Euclidean inner product. For a vector \mathbf{v} denote the ℓ_2 vector norm as $\|\mathbf{v}\|_2$ and for a positive definite matrix \mathbf{S} define the induced vector norm $\|\mathbf{v}\|_{\mathbf{S}} := \sqrt{\mathbf{v}^\top \mathbf{S} \mathbf{v}}$. The projections of \mathbf{x} onto \mathbf{u} and \mathbf{u}^\perp are defined as $\mathbf{x}^{\mathbf{u}} := \frac{\mathbf{u}\mathbf{u}^\top \mathbf{x}}{\|\mathbf{u}\|_2^2}, \mathbf{x}^{\mathbf{u}^\perp} := \left(\mathbf{I} - \frac{\mathbf{u}\mathbf{u}^\top}{\|\mathbf{u}\|_2^2}\right)\mathbf{x}$. Denote the indicator function of event A as $\mathbb{1}_A$ and for a weight vector at time t , $\mathbf{v}_k(t)$, and data point \mathbf{x}_i we denote $\mathbb{1}_{ik}(t) := \mathbb{1}_{\{\mathbf{v}_k(t)^\top \mathbf{x}_i \geq 0\}}$.

WN procedure For a single neuron $\sigma(\mathbf{w}^\top \mathbf{x})$, WN re-parametrizes the weight $\mathbf{w} \in \mathbb{R}^d$ in terms of $\mathbf{v} \in \mathbb{R}^d, g \in \mathbb{R}$ as

$$\mathbf{w}(\mathbf{v}, g) = g \cdot \frac{\mathbf{v}}{\|\mathbf{v}\|_2}, \quad \sigma\left(g \cdot \frac{\mathbf{v}^\top \mathbf{x}}{\|\mathbf{v}\|_2}\right). \quad (3.3)$$

This decouples the magnitude and direction of each weight vector (referred as the “length-direction” decomposition). In comparison, for BN each output $\mathbf{w}^\top \mathbf{x}$ is normalized according

to the average statistics in a batch. We can draw the following analogy between WN and BN if the inputs \mathbf{x}_i are centered ($\mathbb{E}\mathbf{x} = \mathbf{0}$) and the covariance matrix is known ($\mathbb{E}\mathbf{x}\mathbf{x}^\top = \mathbf{S}$). In this case, batch training with BN amounts to

$$\begin{aligned} \sigma\left(\gamma \cdot \frac{\mathbf{w}^\top \mathbf{x}}{\sqrt{\mathbb{E}_{\mathbf{x}}(\mathbf{w}^\top \mathbf{x} \mathbf{x}^\top \mathbf{w})}}\right) &= \sigma\left(\gamma \cdot \frac{\mathbf{w}^\top \mathbf{x}}{\sqrt{\mathbf{w}^\top \mathbf{S} \mathbf{w}}}\right) \\ &= \sigma\left(\gamma \cdot \frac{\mathbf{w}^\top \mathbf{x}}{\|\mathbf{w}\|_{\mathbf{S}}}\right). \end{aligned} \quad (3.4)$$

From this prospective, WN is a special case of (3.4) with $\mathbf{S} = \mathbf{I}$ [SK16, KDL19].

Properties of WN We start by giving an overview of known properties of WN that will be used to derive the gradient flow dynamics of WN training.

For re-parametrization (3.3) of a network function f that is initially parameterized with a weight \mathbf{w} , the gradient $\nabla_{\mathbf{w}} f$ relates to the gradients $\nabla_{\mathbf{v}} f$, $\frac{\partial f}{\partial g}$ by the identities

$$\nabla_{\mathbf{v}} f = \frac{g}{\|\mathbf{v}\|_2} (\nabla_{\mathbf{w}} f)^{\mathbf{v}^\perp}, \quad \frac{\partial f}{\partial g} = (\nabla_{\mathbf{w}} f)^{\mathbf{v}}.$$

This implies that $\nabla_{\mathbf{v}} f \cdot \mathbf{v} = 0$ for each input \mathbf{x} and parameter \mathbf{v} . For gradient flow, this orthogonality results in $\|\mathbf{v}(0)\|_2 = \|\mathbf{v}(t)\|_2$ for all t . For gradient descent (with step size η) the discretization in conjunction with orthogonality leads to increasing parameter magnitudes during training [ALL19b, HBG18, SK16], as illustrated in Figure 3.1,

$$\|\mathbf{v}(s+1)\|_2^2 = \|\mathbf{v}(s)\|_2^2 + \eta^2 \|\nabla_{\mathbf{v}} f\|_2^2 \geq \|\mathbf{v}(s)\|_2^2. \quad (3.5)$$

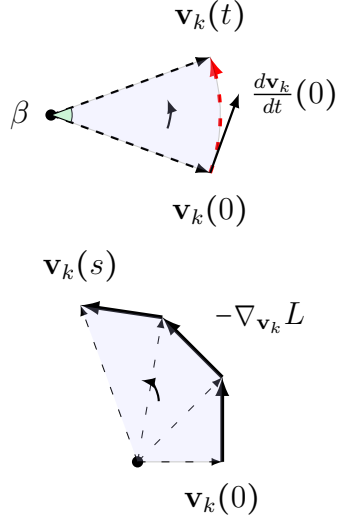


Figure 3.1: WN updates for gradient flow and gradient descent. For gradient flow, the norm of the weights are preserved, i.e., $\|\mathbf{v}_k(0)\|_2 = \|\mathbf{v}_k(t)\|_2$ for all $t > 0$. For gradient descent, the norm of the weights $\|\mathbf{v}_k(s)\|_2$ is increasing with s .

Problem setup We analyze (3.1) with WN training (3.2), so that

$$f(\mathbf{x}; \mathbf{V}, \mathbf{c}, \mathbf{g}) = \frac{1}{\sqrt{m}} \sum_{k=1}^m c_k \sigma \left(g_k \cdot \frac{\mathbf{v}_k^\top \mathbf{x}}{\|\mathbf{v}_k\|_2} \right).$$

We take an initialization in the spirit of [SK16]:

$$\begin{aligned} \mathbf{v}_k(0) &\sim N(0, \beta^2 \mathbf{I}), \quad c_k \sim U\{-1, 1\}, \\ \text{and } g_k(0) &= \|\mathbf{v}_k(0)\|_2 / \beta. \end{aligned} \tag{3.6}$$

Where β^2 is the variance of \mathbf{v}_k at initialization. The initialization of $g_k(0)$ is therefore taken to be independent of β . We remark that the initialization (3.6) gives the same initial output distribution as in methods that study the un-normalized network class (3.1). The parameters of the network are optimized using the training data $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ with respect to the square loss

$$L(f) = \frac{1}{2} \sum_{i=1}^n (f(\mathbf{x}_i) - y_i)^2 = \frac{1}{2} \|\mathbf{f} - \mathbf{y}\|_2^2, \tag{3.7}$$

where $\mathbf{f} = (f_1, \dots, f_n)^\top = (f(\mathbf{x}_1), \dots, f(\mathbf{x}_n))^\top$ and $\mathbf{y} = (y_1, \dots, y_n)^\top$.

3.4 Evolution dynamics

We present the gradient flow dynamics of training (3.7) to illuminate the modified dynamics of WN as compared with vanilla gradient descent. In Appendix 3.A.3 we tackle gradient descent training with WN where the predictions' evolution vector $\frac{d\mathbf{f}}{dt}$ is replaced by the finite difference $\mathbf{f}(s+1) - \mathbf{f}(s)$. For gradient flow, each parameter is updated in the negative direction of the partial derivative of the loss with respect to that parameter. The optimization dynamics give

$$\frac{d\mathbf{v}_k}{dt} = -\frac{\partial L}{\partial \mathbf{v}_k}, \quad \frac{dg_k}{dt} = -\frac{\partial L}{\partial g_k}. \quad (3.8)$$

We consider the case where we fix the top layer parameters c_k during training. In the over-parameterized settings we consider, the dynamics of c_k and g_k turn out to be equivalent. To quantify convergence, we monitor the time derivative of the i -th prediction, which is computed via the chain rule as

$$\frac{\partial f_i}{\partial t} = \sum_{k=1}^m \frac{\partial f_i}{\partial \mathbf{v}_k} \frac{d\mathbf{v}_k}{dt} + \frac{\partial f_i}{\partial g_k} \frac{dg_k}{dt}.$$

Substituting (3.8) into the i -th prediction evolution and grouping terms yields

$$\frac{\partial f_i}{\partial t} = -\underbrace{\sum_{k=1}^m \frac{\partial f_i}{\partial \mathbf{v}_k} \frac{\partial L}{\partial \mathbf{v}_k}}_{T_{\mathbf{v}}^i} - \underbrace{\sum_{k=1}^m \frac{\partial f_i}{\partial g_k} \frac{\partial L}{\partial g_k}}_{T_g^i}. \quad (3.9)$$

The gradients of f_i and L with respect to \mathbf{v}_k are written explicitly as

$$\begin{aligned} \frac{\partial f_i}{\partial \mathbf{v}_k}(t) &= \frac{1}{\sqrt{m}} \frac{c_k \cdot g_k(t)}{\|\mathbf{v}_k(t)\|_2} \cdot \mathbf{x}_i^{\mathbf{v}_k(t)^\perp} \mathbf{1}_{ik}(t), \\ \frac{\partial L}{\partial \mathbf{v}_k}(t) &= \frac{1}{\sqrt{m}} \sum_{i=1}^n (f_i(t) - y_i) \frac{c_k \cdot g_k(t)}{\|\mathbf{v}_k(t)\|_2} \mathbf{x}_i^{\mathbf{v}_k(t)^\perp} \mathbf{1}_{ik}(t). \end{aligned}$$

Defining the \mathbf{v} -orthogonal Gram matrix $\mathbf{V}(t)$ as

$$\begin{aligned} \mathbf{V}_{ij}(t) &= \\ \frac{1}{m} \sum_{k=1}^m \left(\frac{\beta c_k \cdot g_k(t)}{\|\mathbf{v}_k(t)\|_2} \right)^2 \langle \mathbf{x}_i^{\mathbf{v}_k(t)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(t)^\perp} \rangle \mathbf{1}_{ik}(t) \mathbf{1}_{jk}(t), \end{aligned} \quad (3.10)$$

we can compute $T_{\mathbf{v}}^i$ as

$$T_{\mathbf{v}}^i(t) = \sum_{j=1}^n \frac{\mathbf{V}_{ij}(t)}{\beta^2} (f_j(t) - y_j).$$

Note that $\mathbf{V}(t)$ is the induced neural tangent kernel [JGH18] for the parameters \mathbf{v} of WN training. While it resembles the Gram matrix $\mathbf{H}(t)$ studied in [ADH19], here we obtain a matrix that is not piece-wise constant in \mathbf{v} since the data-points are projected onto the orthogonal component of \mathbf{v} . We compute $T_{\mathbf{g}}^i$ in (3.9) analogously. The associated derivatives with respect to g_k are

$$\begin{aligned} \frac{\partial f_i}{\partial g_k}(t) &= \frac{1}{\sqrt{m}} \frac{c_k}{\|\mathbf{v}_k(t)\|_2} \sigma(\mathbf{v}_k(t)^\top \mathbf{x}_i), \\ \frac{\partial L}{\partial g_k}(t) &= \frac{1}{\sqrt{m}} \sum_{j=1}^n (f_j(t) - y_j) \frac{c_k}{\|\mathbf{v}_k(t)\|_2} \sigma(\mathbf{v}_k(t)^\top \mathbf{x}_j), \end{aligned}$$

and we obtain

$$\begin{aligned} T_{\mathbf{g}}^i(t) &= \\ &= \sum_{k=1}^m \frac{1}{m} \sum_{j=1}^n \frac{c_k^2 (f_j(t) - y_j)}{\|\mathbf{v}_k(t)\|_2^2} \sigma(\mathbf{v}_k(t)^\top \mathbf{x}_j) \sigma(\mathbf{v}_k(t)^\top \mathbf{x}_i). \end{aligned}$$

Given that $c_k^2 = 1$, define $\mathbf{G}(t)$ as

$$\mathbf{G}_{ij}(t) = \frac{1}{m} \sum_{k=1}^m \frac{\sigma(\mathbf{v}_k(t)^\top \mathbf{x}_i) \sigma(\mathbf{v}_k(t)^\top \mathbf{x}_j)}{\|\mathbf{v}_k(t)\|_2^2} \quad (3.11)$$

hence we can write

$$T_{\mathbf{g}}^i(t) = \sum_{j=1}^n \mathbf{G}_{ij}(t) (f_j(t) - y_j).$$

Combining $T_{\mathbf{v}}$ and $T_{\mathbf{g}}$, the full evolution dynamics are given by

$$\frac{d\mathbf{f}}{dt} = - \left(\frac{\mathbf{V}(t)}{\beta^2} + \mathbf{G}(t) \right) (\mathbf{f}(t) - \mathbf{y}). \quad (3.12)$$

Denote $\mathbf{\Lambda}(t) := \frac{\mathbf{V}(t)}{\beta^2} + \mathbf{G}(t)$ and write $\frac{d\mathbf{f}}{dt} = -\mathbf{\Lambda}(t)(\mathbf{f}(t) - \mathbf{y})$. We note that $\mathbf{V}(0), \mathbf{G}(0)$, defined in (3.10), (3.11), are independent of β :

Observation 1 (β independence). *For initialization (3.6) and $\beta > 0$ the Gram matrices $\mathbf{V}(0), \mathbf{G}(0)$ are independent of β .*

This fact is proved in Appendix 3.A.1. When training the neural network in (3.1) without WN (see [DZP19, ADH19, ZMG19]), the corresponding neural tangent kernel $\mathbf{H}(t)$ is defined by $\frac{\partial f_i}{\partial t} = \sum_{k=1}^m \frac{\partial f_i}{\partial \mathbf{w}_k} \frac{d\mathbf{w}_k}{dt} = -\sum_{k=1}^m \frac{\partial f_i}{\partial \mathbf{w}_k} \frac{\partial L}{\partial \mathbf{w}_k} = -\sum_{j=1}^n \mathbf{H}_{ij}(t)(f_j - y_j)$ and takes the form

$$\mathbf{H}_{ij}(t) = \frac{1}{m} \sum_{k=1}^m \mathbf{x}_i^\top \mathbf{x}_j \mathbf{1}_{ik}(t) \mathbf{1}_{jk}(t). \quad (3.13)$$

The analysis presented above shows that vanilla and WN gradient descent are related as follows.

Proposition 7. *Define $\mathbf{V}(0)$, $\mathbf{G}(0)$, and $\mathbf{H}(0)$ as in (3.10), (3.11), and (3.13) respectively. then for all $\beta > 0$,*

$$\mathbf{V}(0) + \mathbf{G}(0) = \mathbf{H}(0).$$

Thus, for $\beta = 1$,

$$\frac{\partial \mathbf{f}}{\partial t} = -\mathbf{\Lambda}(0)(\mathbf{f}(0) - \mathbf{y}) = -\mathbf{H}(0)(\mathbf{f}(0) - \mathbf{y}).$$

That is, WN decomposes the NTK in each layer into a length and a direction component. We refer to this as the ‘‘length-direction decoupling’’ of the NTK, in analogy to (3.3). From the proposition, normalized and un-normalized training kernels initially coincide if $\beta = 1$. We hypothesize that the utility of normalization methods can be attributed to the modified NTK $\mathbf{\Lambda}(t)$ that occurs when the WN coefficient, β , deviates from 1. For $\beta \gg 1$ the kernel $\mathbf{\Lambda}(t)$ is dominated by $\mathbf{G}(t)$, and for $\beta \ll 1$ the kernel $\mathbf{\Lambda}(t)$ is dominated by $\mathbf{V}(t)$. We elaborate on the details of this in the next section. In our analysis we will study the two regimes $\beta > 1$ and $\beta < 1$ in turn.

3.5 Main convergence theory

In this section we discuss our convergence theory and main results. From the continuous flow (3.12), we observe that the convergence behavior is described by $\mathbf{V}(t)$ and $\mathbf{G}(t)$. The matrices $\mathbf{V}(t)$ and $\mathbf{G}(t)$ are positive semi-definite since they can be shown to be covariance

matrices. This implies that the least eigenvalue of the evolution matrix $\mathbf{\Lambda}(t) = \frac{1}{\beta^2}\mathbf{V}(t) + \mathbf{G}(t)$ is bounded below by the least eigenvalue of each kernel matrix,

$$\lambda_{\min}(\mathbf{\Lambda}(t)) \geq \max\{\lambda_{\min}(\mathbf{V}(t))/\beta^2, \lambda_{\min}(\mathbf{G}(t))\}.$$

For finite-step gradient descent, a discrete analog of evolution (3.12) holds. However, the discrete case requires additional care in ensuring dominance of the driving gradient terms. For gradient flow, it is relatively easy to see linear convergence is attained by relating the rate of change of the loss to the magnitude of the loss. Suppose that for all $t \geq 0$,

$$\lambda_{\min}(\mathbf{\Lambda}(t)) \geq \omega/2, \quad \text{with } \omega > 0. \quad (3.14)$$

Then the change in the regression loss is written as

$$\begin{aligned} \frac{d}{dt} \|\mathbf{f}(t) - \mathbf{y}\|_2^2 &= 2(\mathbf{f}(t) - \mathbf{y})^\top \frac{d\mathbf{f}(t)}{dt} \\ &= -2(\mathbf{f}(t) - \mathbf{y})^\top \mathbf{\Lambda}(t)(\mathbf{f}(t) - \mathbf{y}) \\ &\stackrel{(3.14)}{\leq} -\omega \|\mathbf{f}(t) - \mathbf{y}\|_2^2. \end{aligned}$$

Integrating this time derivative and using the initial conditions yields

$$\|\mathbf{f}(t) - \mathbf{y}\|_2^2 \leq \exp(-\omega t) \|\mathbf{f}(0) - \mathbf{y}\|_2^2,$$

which gives linear convergence. The focus of our proof is therefore showing that (3.14) holds throughout training.

By Observation 1 we have that \mathbf{V} and \mathbf{G} are independent of the WN coefficient β (β only appears in the $1/\beta^2$ scaling of $\mathbf{\Lambda}$). This suggests that the kernel $\mathbf{\Lambda}(t) = \frac{1}{\beta^2}\mathbf{V}(t) + \mathbf{G}(t)$ can be split into two regimes: When $\beta < 1$ the kernel is dominated by the first term $\frac{1}{\beta^2}\mathbf{V}$, and when $\beta > 1$ the kernel is dominated by the second term \mathbf{G} . We divide our convergence result based on these two regimes.

In each regime, (3.14) holds if the corresponding dominant kernel, $\mathbf{V}(t)$ or $\mathbf{G}(t)$, maintains a positive least eigenvalue. Having a least eigenvalue that is bounded from 0 gives a convex-like property that allows us to prove convergence. To ensure that condition (3.14) is satisfied,

for each regime we show that the corresponding dominant kernel is “anchored” (remains close) to an auxiliary Gram matrix which we define in the following for \mathbf{V} and \mathbf{G} .

Define the auxiliary \mathbf{v} -orthogonal and \mathbf{v} -aligned Gram matrices $\mathbf{V}^\infty, \mathbf{G}^\infty$ as

$$\mathbf{V}_{ij}^\infty := \mathbb{E}_{\mathbf{v} \sim N(0, \beta^2 \mathbf{I})} \langle \mathbf{x}_i^{\mathbf{v}^\perp}, \mathbf{x}_j^{\mathbf{v}^\perp} \rangle \mathbb{1}_{ik}(0) \mathbb{1}_{jk}(0), \quad (3.15)$$

$$\mathbf{G}_{ij}^\infty := \mathbb{E}_{\mathbf{v} \sim N(0, \beta^2 \mathbf{I})} \langle \mathbf{x}_i^{\mathbf{v}}, \mathbf{x}_j^{\mathbf{v}} \rangle \mathbb{1}_{ik}(0) \mathbb{1}_{jk}(0). \quad (3.16)$$

For now, assume that \mathbf{V}^∞ and \mathbf{G}^∞ are positive definite with a least eigenvalue bounded below by ω (we give a proof sketch below). In the convergence proof we will utilize over-parametrization to ensure that $\mathbf{V}(t), \mathbf{G}(t)$ concentrate to their auxiliary versions so that they are also positive definite with a least eigenvalue that is greater than $\omega/2$. The precise formulations are presented in Lemmas 3.A.6 and 3.A.7 that are relegated to Appendix 3.A.2.

To prove our convergence results we make the assumption that the \mathbf{x}_i s have bounded norm and are not parallel.

Assumption 1 (Normalized non-parallel data). *The data points $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)$ satisfy $\|\mathbf{x}_i\|_2 \leq 1$ and for each index pair $i \neq j$, $\mathbf{x}_i \neq \kappa \cdot \mathbf{x}_j$ for all $\kappa \in \mathbb{R} \setminus \{0\}$.*

In order to simplify the presentation of our results, we assume that the input dimension d is not too small, whereby $d \geq 50$ suffices. This is not essential for the proof. Specific details are provided in Appendix 3.A.1.

Assumption 2. *For data $\mathbf{x}_i \in \mathbb{R}^d$ assume that $d \geq 50$.*

Both assumptions can be easily satisfied by pre-processing, e.g., normalizing and shifting the data, and adding zero coordinates if needed.

Given Assumption 1, $\mathbf{V}^\infty, \mathbf{G}^\infty$ are shown to be positive definite.

Lemma 3.5.1. *Fix training data $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ satisfying Assumption 1. Then the \mathbf{v} -orthogonal and \mathbf{v} -aligned Gram matrices \mathbf{V}^∞ and \mathbf{G}^∞ , defined as in (3.15) and (3.16), are strictly positive definite. We denote the least eigenvalues $\lambda_{\min}(\mathbf{V}^\infty) =: \lambda_0$, $\lambda_{\min}(\mathbf{G}^\infty) =: \mu_0$.*

Proof sketch Here we sketch the proof of Lemma 3.5.1. The main idea, is the same as [DZP19], is to regard the auxiliary matrices $\mathbf{V}^\infty, \mathbf{G}^\infty$ as the covariance matrices of linearly independent operators. For each data point \mathbf{x}_i , define $\phi_i(\mathbf{v}) := \mathbf{x}_i^{\mathbf{y}} \mathbb{1}_{\{\mathbf{x}_i^\top \mathbf{v} \geq 0\}}$. The Gram matrix \mathbf{V}^∞ is the covariance matrix of $\{\phi_i\}_{i=1:n}$ taken over \mathbb{R}^d with the measure $N(0, \beta^2 \mathbf{I})$. Hence showing that \mathbf{V}^∞ is strictly positive definite is equivalent to showing that $\{\phi_i\}_{i=1,\dots,n}$ are linearly independent. Unlike [DZP19], the functionals under consideration are not piecewise constant so a different construction is used to prove independence. Analogously, a new set of operators, $\theta_i(\mathbf{v}) := \sigma(\mathbf{x}_i^{\mathbf{y}})$, is constructed for \mathbf{G}^∞ . Interestingly, each ϕ_i corresponds to $\frac{d\theta_i}{d\mathbf{v}}$. The full proof is presented in Appendix 3.A.2.2. As already observed from evolution (3.12), different magnitudes of β can lead to two distinct regimes that are discussed below. We present the main results for each regime.

V-dominated convergence

For $\beta < 1$ convergence is dominated by $\mathbf{V}(t)$ and $\lambda_{\min}(\mathbf{\Lambda}(t)) \geq \frac{1}{\beta^2} \lambda_{\min}(\mathbf{V}(t))$. We present the convergence theorem for the \mathbf{V} -dominated regime here.

Theorem 3.5.2 (V-dominated convergence). *Suppose a neural network of the form (3.2) is initialized as in (3.6) with $\beta \leq 1$ and that Assumptions 1,2 hold. In addition, suppose the neural network is trained via the regression loss (3.7) with targets \mathbf{y} satisfying $\|\mathbf{y}\|_\infty = O(1)$. If $m = \Omega(n^4 \log(n/\delta)/\lambda_0^4)$, then with probability $1 - \delta$,*

1. For iterations $s = 0, 1, \dots$, the evolution matrix $\mathbf{\Lambda}(s)$ satisfies $\lambda_{\min}(\mathbf{\Lambda}(s)) \geq \frac{\lambda_0}{2\beta^2}$.
2. WN training with gradient descent of step-size $\eta = O\left(\frac{\beta^2}{\|\mathbf{V}^\infty\|_2}\right)$ converges linearly as

$$\|\mathbf{f}(s) - \mathbf{y}\|_2^2 \leq \left(1 - \frac{\eta\lambda_0}{2\beta^2}\right)^s \|\mathbf{f}(0) - \mathbf{y}\|_2^2.$$

The proof of Theorem 3.5.2 is presented in Appendix 3.A.3. We will provide a sketch below. We make the following observations about our \mathbf{V} -dominated convergence result.

The required over-parametrization m is independent of β . Further, the dependence of m

on the failure probability is $\log(1/\delta)$. This improves previous results that require polynomial dependence of order δ^3 . Additionally, we reduce the dependence on the sample size from n^6 (as appears in [ADH19]) to $n^4 \log(n)$.

In Theorem 3.5.2, smaller β leads to faster convergence, since the convergence is dictated by λ_0/β^2 . Nonetheless, smaller β is also at the cost of smaller allowed step-sizes, since $\eta = O(\beta^2/\|\mathbf{V}^\infty\|_2)$. The trade-off between step-size and convergence speed is typical. For example, this is implied in Chizat et al. [COB19], where nonetheless the authors point out that for gradient flow training, the increased convergence rate is not balanced by a limitation on the step-size. The works [HBG18, WWB18, ALL19b] define an effective step-size (adaptive step-size) $\eta' = \eta/\beta^2$ to avoid the dependence of η on β .

G-dominated convergence

For $\beta > 1$ our convergence result for the class (3.2) is based on monitoring the least eigenvalue of $\mathbf{G}(t)$. Unlike \mathbf{V} -dominated convergence, β does not affect the convergence speed in this regime.

Theorem 3.5.3 (**G-dominated convergence**). *Suppose a network of the form (3.2) is initialized as in (3.6) with $\beta \geq 1$ and that Assumptions 1, 2 hold. In addition, suppose the neural network is trained via the regression loss (3.7) with targets \mathbf{y} satisfying $\|\mathbf{y}\|_\infty = O(1)$. If $m = \Omega(\max\{n^4 \log(n/\delta)/\beta^4 \mu_0^4, n^2 \log(n/\delta)/\mu_0^2\})$, then with probability $1 - \delta$,*

1. For iterations $s = 0, 1, \dots$, the evolution matrix $\mathbf{\Lambda}(s)$ satisfies $\lambda_{\min}(\mathbf{\Lambda}(s)) \geq \frac{\mu_0}{2}$.
2. WN training with gradient descent of step-size $\eta = O\left(\frac{1}{\|\mathbf{\Lambda}(t)\|}\right)$ converges linearly as

$$\|\mathbf{f}(s) - \mathbf{y}\|_2^2 \leq \left(1 - \frac{\eta \mu_0}{2}\right)^s \|\mathbf{f}(0) - \mathbf{y}\|_2^2.$$

We make the following observations about our \mathbf{G} -dominated convergence result, and provide a proof sketch further below.

Theorem 3.5.3 holds for $\beta \geq 1$ so long as $m = \Omega(\max\{n^4 \log(n/\delta)/\mu_0^4 \beta^4, n^2 \log(n/\delta)/\mu_0^2\})$. Taking $\beta = \sqrt{n/\mu_0}$ gives an optimal required over-parametrization of order $m = \Omega(n^2 \log(n/\delta)/\mu_0^2)$. This significantly improves on previous results [DZP19] for un-normalized training that have dependencies of order 4 in the least eigenvalue, cubic dependence in $1/\delta$, and n^6 dependence in the number of samples n . In contrast to \mathbf{V} -dominated convergence, here the rate of convergence μ_0 is independent of β but the over-parametrization m is β -dependent. We elaborate on this curious behavior in the next sections.

Proof sketch of main results The proof of Theorems 3.5.2 and 3.5.3 is inspired by a series of works including [DZP19, ADH19, ZMG19, WDW19, DLL19a]. The proof has the following steps: **(I)** We show that at initialization $\mathbf{V}(0), \mathbf{G}(0)$ can be viewed as empirical estimates of averaged data-dependent kernels $\mathbf{V}^\infty, \mathbf{G}^\infty$ that are strictly positive definite under Assumption 1. **(II)** For each regime, we prove that the corresponding kernel remains positive definite if $\mathbf{v}_k(t)$ and $g_k(t)$ remain near initialization for each $1 \leq k \leq m$. **(III)** Given a uniformly positive definite evolution matrix $\mathbf{\Lambda}(t)$ and sufficient over-parametrization we show that each neuron, $\mathbf{v}_k(t), g_k(t)$ remains close to its initialization. The full proof is presented in Appendix 3.A.2 for gradient flow and Appendix 3.A.3 for finite-step gradient descent. Next we interpret the main results and discuss how the modified NTK in WN can be viewed as a form of natural gradient.

Connection with natural gradient Natural gradient methods define the steepest descent direction in the parameter space of a model from the perspective of function space. This amounts to introducing a particular geometry into the parameter space which is reflective of the geometry of the corresponding functions. A re-parametrization of a model, and WN in particular, can also be interpreted as choosing a particular geometry for the parameter space. This gives us a perspective from which to study the effects of WN. The recent work of [ZMG19] studies the effects of natural gradient methods from the lens of the NTK and shows that when optimizing with the natural gradient, one is able to get significantly improved

training speed. In particular, using the popular natural gradient method K-FAC improves the convergence speed considerably.

Natural gradients transform the NTK from $\mathbf{J}\mathbf{J}^\top$ to $\mathbf{J}\mathbf{G}^\dagger\mathbf{J}^\top$, where \mathbf{J} is the Jacobian with respect to the parameters and \mathbf{G} is the metric. The WN re-parametrization transforms the NTK from $\mathbf{J}\mathbf{J}^\top$ to $\mathbf{J}\mathbf{S}^\top\mathbf{S}\mathbf{J}^\top$. To be more precise, denote the un-normalized NTK as $\mathbf{H} = \mathbf{J}\mathbf{J}^\top$, where \mathbf{J} is the Jacobian matrix for $\mathbf{x}_1, \dots, \mathbf{x}_n$ written in a compact tensor as $\mathbf{J} = [\mathbf{J}_1, \dots, \mathbf{J}_n]^\top$ with $\mathbf{J}_i = \left[\frac{\partial f(\mathbf{x}_i)}{\partial \mathbf{w}_1} \dots \frac{\partial f(\mathbf{x}_i)}{\partial \mathbf{w}_m} \right]$, where matrix multiplication is a slight abuse of notation. Namely $\mathbf{J} \in \mathbb{R}^{n \times m \times d}$ and we define multiplication of $\mathbf{A} \in \mathbb{R}^{n \times m \times d} \times \mathbf{B} \in \mathbb{R}^{d \times m \times p} \rightarrow \mathbf{AB} \in \mathbb{R}^{n \times p}$ as

$$(\mathbf{AB})_{ij} = \sum_{k=1}^m \langle \mathbf{A}_{ik:}, \mathbf{B}_{:kj} \rangle.$$

For any re-parametrization $\mathbf{w}(\mathbf{r})$, we have that

$$\mathbf{\Lambda} = \mathbf{K}\mathbf{K}^\top,$$

where $\mathbf{K} = \mathbf{J}\mathbf{S}^\top$ and \mathbf{S} corresponds to the Jacobian of the re-parametrization $\mathbf{w}(\mathbf{r})$. By introducing WN layers the reparameterized NTK is compactly written as

$$\mathbf{\Lambda} = \mathbf{J}\mathbf{S}^\top\mathbf{S}\mathbf{J}^\top.$$

Here $\mathbf{S} = [\mathbf{S}_1, \dots, \mathbf{S}_m]$ with

$$\mathbf{S}_k = \left[\frac{g_k}{\|\mathbf{v}_k\|_2} \left(\mathbf{I} - \frac{\mathbf{v}_k \mathbf{v}_k^\top}{\|\mathbf{v}_k\|_2} \right), \frac{\mathbf{v}_k}{\|\mathbf{v}_k\|_2} \right].$$

The term $\mathbf{N}(\beta) := \mathbf{S}\mathbf{S}^\top$ leads to a family of different gradient re-parametrizations depending on β . The above representation of the WN NTK is equivalent to $\mathbf{\Lambda}(\beta) = \frac{1}{\beta^2} \mathbf{V} + \mathbf{G} = \mathbf{J}\mathbf{N}(\beta)\mathbf{J}^\top$. For different initialization magnitudes β , $\mathbf{N}(\beta)$ leads to different NTKs with modified properties.

For $\beta = 1$ the term corresponds to training without normalization, yet over $\beta \in (0, \infty)$, $\mathbf{N}(\beta)$ leads to a family NTKs with different properties. In addition there exists an β^* that maximizes the convergence rate. Such β^* is either a proper global maximum or is attained at one of $\beta \rightarrow 0, \beta \rightarrow \infty$. For the latter, one may fix β^* with $\beta^* \ll 1$ or $\beta^* \gg 1$ respectively so that there exists β^* that outpaces un-normalized convergence ($\beta = 1$). This leads to equal or faster convergence of WN as compared with un-normalized training:

Proposition 8 (Fast Convergence of WN). *Suppose a neural network of the form (3.2) is initialized as in (3.6) and that Assumptions 1,2 hold. In addition, suppose the network is trained via the regression loss (3.7) with targets \mathbf{y} satisfying $\|\mathbf{y}\|_\infty = O(1)$. Then, with probability $1 - \delta$ over the initialization, there exists β^* such that WN training with β^* initialization leads to faster convergence: If $m = \Omega(n^4 \log(n/\delta) / \min\{\lambda_0^4, \mu_0^4\})$,*

1. *WN training with gradient descent of step-size $\eta_{\beta^*} = O\left(\frac{1}{\|\mathbf{v}^\infty/(\beta^*)^2 + \mathbf{G}^\infty\|_2}\right)$ converges linearly as*

$$\begin{aligned} \|\mathbf{f}(s) - \mathbf{y}\|_2^2 &\leq \\ &\left(1 - \eta_{\beta^*}(\lambda_0/2(\beta^*)^2 + \mu_0/2)\right)^s \|\mathbf{f}(0) - \mathbf{y}\|_2^2. \end{aligned}$$

2. *The convergence rate of WN is faster than un-normalized convergence,*

$$(1 - \eta_{\beta^*} \lambda_{\min}(\mathbf{\Lambda}(s))) \leq (1 - \eta \lambda_{\min}(\mathbf{H}(s))).$$

This illustrates the utility of WN from the perspective of the NTK, guaranteeing that there exists an β^* that leads to faster convergence in *finite-step* gradient descent as compared with un-normalized training.

3.6 Discussion

Dynamic normalization is the most common optimization set-up of current deep learning models, yet understanding the convergence of such optimization methods is still an open problem. This chapter presents a proof giving sufficient conditions for convergence of dynamically normalized 2-layer ReLU networks trained with gradient descent. To the best of our knowledge this is the first proof showcasing convergence of gradient descent training of neural networks with dynamic normalization and general data, where the objective function is non-smooth and non-convex. To understand the canonical behavior of each normalization layer, we study the shallow neural network case, that enables us to focus on a single layer

and illustrate the dynamics of weight normalization. Nonetheless, using the techniques presented in [ALS19, DLL19a] we believe that the proofs can be extended to deep networks as a future direction. Through our analysis notion of “length-direction decoupling” is clarified by the neural tangent kernel $\mathbf{\Lambda}(t)$ that naturally separates in our analysis into “length”, $\mathbf{G}(t)$, and “direction”, $\mathbf{V}(t)/\beta^2$, components. For $\beta = 1$ the decomposition initially matches un-normalized training. Yet it is shown that in general, normalized training with gradient descent leads to 2 regimes dominated by different pieces of the neural tangent kernel. The improved analysis reduces the amount of over-parametrization that was needed in previous convergence works in the un-normalized setting and in the \mathbf{G} -dominated regime, we prove convergence with a significantly lower amount of over-parametrization as compared with un-normalized training.

3.A Appendix

We present the detailed proofs of the main results of the chapter below. The appendix is organized as follows. We provide proofs to the simple propositions regarding the NTK presented in the chapter in Sub-appendix 3.A.1, and prove the main results for \mathbf{V} -dominated and \mathbf{G} -dominated convergence in the settings of gradient flow and gradient descent in the beginning of Sub-appendices 3.A.2 and 3.A.3. The proofs for gradient flow and gradient descent share the same main idea, yet the proof for gradient descent has a considerable number of additional technicalities. In the rest of Sub-appendices 3.A.2, 3.A.3 (3.A.2.2 and (3.A.3.1)) we prove the lemmas used in the analysis of flow and finite-step proofs respectively. Before we move forward we highlight some of the challenges of the WN proof.

Distinctive aspects of the WN convergence analysis The main idea of our proof are familiar and structured similarly to the work by [DZP19] on the un-normalized setting. However, the majority of the proofs are modified significantly to account for WN. To the best of our knowledge, the finite-step analysis that we present in Appendix 3.A.3 is entirely new, incorporating updates of both \mathbf{v} and g . The proof of Theorem 3.A.17 is crucially dependent on the geometry of WN gradient descent and the orthogonality property, in particular (3.5). Updates of the weights in both the numerator and denominator require additional analysis that is presented in Lemma 3.A.12. In Appendix 3.A.3.1 we prove Theorems 3.5.2, 3.5.3 based on the general Theorem 3.A.17 and Property 1 which is based on new detailed decomposition of the finite-step difference between iterations. In contrast to the un-normalized setting, the auxiliary matrices $\mathbf{V}^\infty, \mathbf{G}^\infty$ that we have in the WN analysis are not piece-wise constant in \mathbf{v} . To prove they are positive definite, we prove Lemma 3.5.1 based on two new constructive arguments. We develop the technical Lemma 3.A.13 and utilize Bernstein’s inequality to reduce the amount of required over-parametrization in our final bounds on the width m . The amount of over-parameterization in relation to the sample size n is reduced (from n^6 to n^4) through more careful arguments in Lemmas 3.A.5 and 3.A.6, which introduce an

intermediate matrix $\hat{\mathbf{V}}(t)$ and follow additional geometrical identities. Lemma 3.A.11 reduces the polynomial dependence on the failure probability δ to logarithmic dependence based on sub-Gaussian concentration. The denominator in the WN architecture necessitates worst bound analysis which we handle in Lemma 3.A.12 that is used throughout the proofs.

3.A.1 Weight Normalization dynamics proofs

In this section we provide proofs for Proposition 7, which describes the relation between vanilla and WeightNorm NTKs and Observation 1.

Proof of Proposition 7:

We would like to show that $\mathbf{V}(0) + \mathbf{G}(0) = \mathbf{H}(0)$. For each entry, consider

$$(\mathbf{V}(0) + \mathbf{G}(0))_{ij} = \frac{1}{m} \sum_{k=1}^m \langle \mathbf{x}_i^{\mathbf{v}_k(0)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(0)^\perp} \rangle \mathbb{1}_{ik}(0) \mathbb{1}_{jk}(0) + \frac{1}{m} \sum_{k=1}^m \langle \mathbf{x}_i^{\mathbf{v}_k(0)}, \mathbf{x}_j^{\mathbf{v}_k(0)} \rangle \mathbb{1}_{ik}(0) \mathbb{1}_{jk}(0).$$

Note that

$$\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \langle \mathbf{x}_i^{\mathbf{v}_k(0)^\perp} + \mathbf{x}_i^{\mathbf{v}_k(0)}, \mathbf{x}_j^{\mathbf{v}_k(0)^\perp} + \mathbf{x}_j^{\mathbf{v}_k(0)} \rangle = \langle \mathbf{x}_i^{\mathbf{v}_k(0)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(0)^\perp} \rangle + \langle \mathbf{x}_i^{\mathbf{v}_k(0)}, \mathbf{x}_j^{\mathbf{v}_k(0)} \rangle.$$

This gives

$$(\mathbf{V}(0) + \mathbf{G}(0))_{ij} = \frac{1}{m} \sum_{k=1}^m \langle \mathbf{x}_i, \mathbf{x}_j \rangle \mathbb{1}_{ik}(0) \mathbb{1}_{jk}(0) = \mathbf{H}_{ij}(0)$$

which proves the claim. □

Proof of Observation 1:

We show that the initialization of the network is independent of β . Take $\beta, \kappa > 0$, and for each k , initialize $\mathbf{v}_k^\beta, \mathbf{v}_k^\kappa$ as

$$\mathbf{v}_k^\beta(0) \sim N(0, \beta^2 \mathbf{I}), \quad \mathbf{v}_k^\kappa(0) \sim N(0, \kappa^2 \mathbf{I}).$$

Then

$$\frac{\mathbf{v}_k^\beta(0)}{\|\mathbf{v}_k^\beta(0)\|_2} \sim \frac{\mathbf{v}_k^\kappa(0)}{\|\mathbf{v}_k^\kappa(0)\|_2} \sim \text{Unif}(\mathcal{S}^{d-1}) \quad (\text{in distribution}).$$

Hence the distribution of each neuron $\sigma\left(\frac{\mathbf{v}_k(0)}{\|\mathbf{v}_k(0)\|_2}\right)$ at initialization is independent of β . Next for $g_k(0)$, we note that

$$\|\mathbf{v}_k^\beta(0)\|_2 \sim \frac{\beta}{\kappa} \|\mathbf{v}_k^\kappa(0)\|_2.$$

Initializing $g_k^\beta(0), g_k^\kappa(0)$ as in (3.6),

$$g_k^\beta(0) = \frac{\|\mathbf{v}_k(0)\|_2}{\beta}, \quad g_k^\kappa(0) = \frac{\|\mathbf{v}_k(0)\|_2}{\kappa},$$

gives

$$g_k^\beta(0), \quad g_k^\kappa(0) \sim \chi_d, \quad \text{and} \quad \frac{g_k^\beta(0)\mathbf{v}_k^\beta(0)}{\|\mathbf{v}_k^\beta(0)\|_2} \sim \frac{g_k^\kappa(0)\mathbf{v}_k^\kappa(0)}{\|\mathbf{v}_k^\kappa(0)\|_2} \sim N(0, \mathbf{I}),$$

for all β, κ . This shows that the network initialization is independent of β and is equivalent to the initialization of the un-normalized setting. Similarly, inspecting the terms in the summands of $\mathbf{V}(0), \mathbf{G}(0)$ shows that they are also independent of β . For

$$\mathbf{V}_{ij}(0) = \frac{1}{m} \sum_{k=1}^m \mathbb{1}_{ik}(0) \mathbb{1}_{jk}(0) \left(\frac{\beta c_k \cdot g_k(0)}{\|\mathbf{v}_k(0)\|_2} \right)^2 \langle \mathbf{x}_i^{\mathbf{v}_k(0)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(0)^\perp} \rangle$$

the terms $\mathbb{1}_{ik}(0), \mathbf{x}_i^{\mathbf{v}_k(0)^\perp}$ are independent of scale, and the fraction in the summand is identically 1. $\mathbf{G}(0)$ defined as

$$\mathbf{G}_{ij}(0) = \frac{1}{m} \sum_{k=1}^m \mathbb{1}_{ik}(0) \mathbb{1}_{jk}(0) \langle \mathbf{x}_i^{\mathbf{v}_k(0)}, \mathbf{x}_j^{\mathbf{v}_k(0)} \rangle$$

is also invariant of scale since the projection onto a vector direction $\mathbf{v}_k(0)$ is independent of scale. □

3.A.2 Convergence proof for gradient flow

In this section we derive the convergence results for gradient flow.

The main results are analogous to Theorems 3.5.2, 3.5.3 but by considering gradient flow instead of gradient descent the proofs are simplified. In Appendix 3.A.3 we prove the main results from Section 3.5 (Theorem 3.5.2, 3.5.3) for finite step gradient descent.

We state our convergence results for gradient flow.

Theorem 3.A.1 (**V**-dominated convergence). *Suppose a network from the class (3.2) is initialized as in (3.6) with $\beta < 1$ and that assumptions 1,2 hold. In addition, suppose the neural network is trained via the regression loss (3.7) with target \mathbf{y} satisfying $\|\mathbf{y}\|_\infty = O(1)$. Then if $m = \Omega(n^4 \log(n/\delta)/\lambda_0^4)$, WeightNorm training with gradient flow converges at a linear rate, with probability $1 - \delta$, as*

$$\|\mathbf{f}(t) - \mathbf{y}\|_2^2 \leq \exp(-\lambda_0 t / \beta^2) \|\mathbf{f}(0) - \mathbf{y}\|_2^2.$$

This theorem is analogous to Theorem 3.5.2 but since here, the settings are of gradient flow there is no mention of the step-size. It is worth noting that smaller β leads to faster convergence and appears to not affect the other hypotheses of the flow theorem. This “un-interrupted” fast convergence behavior does not extend to finite-step gradient descent where the increased convergence rate is balanced by decreasing the allowed step-size.

The second main result for gradient flow is for **G**-dominated convergence.

Theorem 3.A.2 (**G**-dominated convergence). *Suppose a network from the class (3.2) is initialized as in (3.6) with $\beta > 1$ and that assumptions 1, 2 hold. In addition, suppose the neural network is trained on the regression loss (3.7) with target \mathbf{y} satisfying $\|\mathbf{y}\|_\infty = O(1)$. Then if $m = \Omega(\max\{n^4 \log(n/\delta)/\beta^4 \mu_0^4, n^2 \log(n/\delta)/\mu_0^2\})$, WeightNorm training with gradient flow converges at a linear rate, with probability $1 - \delta$, as*

$$\|\mathbf{f}(t) - \mathbf{y}\|_2^2 \leq \exp(-\mu_0 t) \|\mathbf{f}(0) - \mathbf{y}\|_2^2.$$

3.A.2.1 Proof sketch

To prove the results above we follow the steps introduced in the proof sketch of Section 3.5. The main idea of the proofs for **V** and **G** dominated convergence are analogous and a lot of the proofs are based of [DZP19]. We show that in each regime, we attain linear convergence by proving that the least eigenvalue of the evolution matrix $\mathbf{\Lambda}(t)$ is strictly positive. For the **V**-dominated regime we lower bound the least eigenvalue of $\mathbf{\Lambda}(t)$ as

$\lambda_{\min}(\mathbf{\Lambda}(t)) \geq \lambda_{\min}(\mathbf{V}(t))/\beta^2$ and in the \mathbf{G} -dominated regime we lower bound the least eigenvalue as $\lambda_{\min}(\mathbf{\Lambda}(t)) \geq \lambda_{\min}(\mathbf{G}(t))$.

The main part of the proof is showing that $\lambda_{\min}(\mathbf{V}(t)), \lambda_{\min}(\mathbf{G}(t))$ stay uniformly positive. We use several lemmas to show this claim.

In each regime, we first show that at initialization the kernel under consideration, $\mathbf{V}(0)$ or $\mathbf{G}(0)$, has a positive least eigenvalue. This is shown via concentration to an auxiliary kernel (Lemmas 3.A.3, 3.A.4), and showing that the auxiliary kernel is also strictly positive definite (Lemma 3.5.1).

Lemma 3.A.3. *Let $\mathbf{V}(0)$ and \mathbf{V}^∞ be defined as in (3.10) and (3.15), assume the network width m satisfies $m = \Omega\left(\frac{n^2 \log(n/\delta)}{\lambda_0^2}\right)$. Then with probability $1 - \delta$,*

$$\|\mathbf{V}(0) - \mathbf{V}^\infty\|_2 \leq \frac{\lambda_0}{4}.$$

Lemma 3.A.4. *Let $\mathbf{G}(0)$ and \mathbf{G}^∞ be defined as in (3.11) and (3.16), assume m satisfies $m = \Omega\left(\frac{n^2 \log(n/\delta)}{\mu_0^2}\right)$. Then with probability $1 - \delta$,*

$$\|\mathbf{G}(0) - \mathbf{G}^\infty\|_2 \leq \frac{\mu_0}{4}.$$

After showing that $\mathbf{V}(0), \mathbf{G}(0)$ have a positive least-eigenvalue we show that $\mathbf{V}(t), \mathbf{G}(t)$ maintain this positive least eigenvalue during training. This part of the proof depends on the over-parametrization of the networks. The main idea is showing that if the individual parameters $\mathbf{v}_k(t), g_k(t)$ do not change too much during training, then $\mathbf{V}(t), \mathbf{G}(t)$ remain close enough to $\mathbf{V}(0), \mathbf{G}(0)$ so that they are still uniformly strictly positive definite. We prove the results for $\mathbf{V}(t)$ and $\mathbf{G}(t)$ separately since each regime imposes different restrictions on the trajectory of the parameters.

For now, in Lemmas 3.A.5, 3.A.6, 3.A.7, we make assumptions on the parameters of the network not changing “too much”; later we show that this holds and is the result of over-parametrization. Specifically, over-parametrization ensures that the parameters stay at a small maximum distance from their initialization.

V-dominated convergence To prove the least eigenvalue condition on $\mathbf{V}(t)$, we introduce the surrogate Gram matrix $\hat{\mathbf{V}}(t)$ defined entry-wise as

$$\hat{\mathbf{V}}_{ij}(t) = \frac{1}{m} \sum_{k=1}^m \langle \mathbf{x}_i^{\mathbf{v}_k(t)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(t)^\perp} \rangle \mathbb{1}_{ik}(t) \mathbb{1}_{jk}(t). \quad (3.17)$$

This definition aligns with $\mathbf{V}(t)$ if we replace the scaling term $\left(\frac{\beta c_k g_k(t)}{\|\mathbf{v}_k(t)\|_2}\right)^2$ in each term in the sum $\mathbf{V}_{ij}(t)$ by 1.

To monitor $\mathbf{V}(t) - \mathbf{V}(0)$ we consider $\hat{\mathbf{V}}(t) - \mathbf{V}(0)$ and $\mathbf{V}(t) - \hat{\mathbf{V}}(t)$ in Lemmas 3.A.5 and 3.A.6 respectively:

Lemma 3.A.5 (Rectifier sign-changes). *Suppose $\mathbf{v}_1(0), \dots, \mathbf{v}_k(0)$ are sampled i.i.d. as (3.6). In addition assume we have $m = \Omega\left(\frac{(m/\delta)^{1/d} n \log(n/\delta)}{\lambda_0}\right)$ and $\|\mathbf{v}_k(t) - \mathbf{v}_k(0)\|_2 \leq \frac{\beta \lambda_0}{96n(m/\delta)^{1/d}} =: R_v$. Then with probability $1 - \delta$,*

$$\|\hat{\mathbf{V}}(t) - \mathbf{V}(0)\|_2 \leq \frac{\lambda_0}{8}.$$

Lemma 3.A.6. *Define*

$$R_g = \frac{\lambda_0}{48n(m/\delta)^{1/d}}, \quad R_v = \frac{\beta \lambda_0}{96n(m/\delta)^{1/d}}. \quad (3.18)$$

Suppose the conditions of Lemma 3.A.5 hold, and that $\|\mathbf{v}_k(t) - \mathbf{v}_k(0)\|_2 \leq R_v$, $\|g_k(t) - g_k(0)\|_2 \leq R_g$ for all $1 \leq k \leq m$. Then with probability $1 - \delta$,

$$\|\mathbf{V}(t) - \mathbf{V}(0)\|_2 \leq \frac{\lambda_0}{4}.$$

G-dominated convergence We ensure that $\mathbf{G}(t)$ stays uniformly positive definite if the following hold.

Lemma 3.A.7. *Given $\mathbf{v}_1(0), \dots, \mathbf{v}_k(0)$ generated i.i.d. as in (3.6), suppose that for each k , $\|\mathbf{v}_k(t) - \mathbf{v}_k(0)\|_2 \leq \frac{\sqrt{2\pi}\beta\mu_0}{8n(m/\delta)^{1/d}} =: \tilde{R}_v$, then with probability $1 - \delta$,*

$$\|\mathbf{G}(t) - \mathbf{G}(0)\|_2 \leq \frac{\mu_0}{4}.$$

After deriving sufficient conditions to maintain a positive least eigenvalue at training, we restate the discussion of linear convergence from Section 3.5 formally.

Lemma 3.A.8. *Consider the linear evolution $\frac{d\mathbf{f}}{dt} = -(\mathbf{G}(t) + \frac{\mathbf{V}(t)}{\beta^2})(\mathbf{f}(t) - \mathbf{y})$ from (3.12). Suppose that $\lambda_{\min}(\mathbf{G}(t) + \frac{\mathbf{V}(t)}{\beta^2}) \geq \frac{\omega}{2}$ for all times $0 \leq t \leq T$. Then*

$$\|\mathbf{f}(t) - \mathbf{y}\|_2^2 \leq \exp(-\omega t) \|\mathbf{f}(0) - \mathbf{y}\|_2^2$$

for all times $0 \leq t \leq T$.

Using the linear convergence result of Lemma 3.A.8, we can now bound the trajectory of the parameters from their initialization.

Lemma 3.A.9. *Suppose that for all $0 \leq t \leq T$, $\lambda_{\min}(\mathbf{G}(t) + \frac{1}{\beta^2}\mathbf{V}(t)) \geq \frac{\omega}{2}$ and $|g_k(t) - g_k(0)| \leq R_g \leq 1/(m/\delta)^{1/d}$. Then with probability $1 - \delta$ over the initialization*

$$\|\mathbf{v}_k(t) - \mathbf{v}_k(0)\|_2 \leq \frac{4\sqrt{n}\|\mathbf{f}(0) - \mathbf{y}\|_2}{\beta\omega\sqrt{m}} =: R'_v \quad (3.19)$$

for each k and all times $0 \leq t \leq T$.

Lemma 3.A.10. *Suppose that for all $0 \leq t \leq T$, $\lambda_{\min}(\mathbf{G}(t) + \frac{1}{\beta^2}\mathbf{V}(t)) \geq \frac{\omega}{2}$. Then with probability $1 - \delta$ over the initialization*

$$|g_k(t) - g_k(0)| \leq \frac{4\sqrt{n}\|\mathbf{f}(0) - \mathbf{y}\|_2}{\sqrt{m}\omega} =: R'_g$$

for each k and all times $0 \leq t \leq T$.

The distance of the parameters from initialization depends on the convergence rate (which depends on $\lambda_{\min}(\mathbf{\Lambda}(t))$) and the width of the network m . We therefore are able to find sufficiently large m for which the maximum parameter trajectories are not too large so that we have that the least eigenvalue of $\mathbf{\Lambda}(t)$ is bounded from 0; this proves the main claim.

Before proving the main results in the case of gradient flow, we use two more technical lemmas.

Lemma 3.A.11. *Suppose that the network is initialized as (3.6) and that $\mathbf{y} \in \mathbb{R}^n$ has bounded entries $|y_i| \leq M$. Then $\|\mathbf{f}(0) - \mathbf{y}\|_2 \leq C\sqrt{n \log(n/\delta)}$ for some absolute constant $C > 0$.*

Lemma 3.A.12 (Failure over initialization). *Suppose $\mathbf{v}_1(0), \dots, \mathbf{v}_k(0)$ are initialized i.i.d. as in (3.6) with input dimension d . Then with probability $1 - \delta$,*

$$\max_{k \in [m]} \frac{1}{\|\mathbf{v}_k(0)\|_2} \leq \frac{(m/\delta)^{1/d}}{\beta}.$$

In addition by (3.5), for all $t \geq 0$, with probability $1 - \delta$,

$$\max_{k \in [m]} \frac{1}{\|\mathbf{v}_k(t)\|_2} \leq \frac{(m/\delta)^{1/d}}{\beta}.$$

Remark (Assumption 2). *Predominately, machine learning applications reside in the high dimensional regime with $d \geq 50$. Typically $d \gg 50$. This therefore leads to an expression $(m/\delta)^{1/d}$ that is essentially constant. For example, if $d = 50$, for $\max_{k \in [m]} \frac{1}{\|\mathbf{v}_k(0)\|_2} \geq 10$, one would need $m/\delta \geq 10^{80}$ (the tail of χ_d^2 also has a factor of $(d/2)! \cdot 2^{d/2}$ which makes the assumption even milder). The term $(m/\delta)^{1/d}$ therefore may be taken as a constant for practicality,*

$$\max_{k \in [m]} \frac{1}{\|\mathbf{v}_k(0)\|_2} \leq \frac{C}{\beta}.$$

While we make Assumption 2 when presenting our final bounds, for transparency we do not use Assumption 2 during our analysis and apply it only when we present the final over-parametrization results to avoid the overly messy bound. Without the assumption the theory still holds yet the over-parametrization bound worsens by a power $1 + 1/(d - 1)$. This is since the existing bounds can be modified, replacing m with $m^{1-\frac{1}{d}}$.

Proof of Theorem 3.A.1:

By substituting $m = \Omega(n^4 \log(n/\delta)/\lambda_0^4)$ and using the bound on $\|\mathbf{f}(0) - \mathbf{y}\|_2$ of Lemma 3.A.11, a direct calculation shows that

$$\|\mathbf{v}_k(t) - \mathbf{v}_k(0)\|_2 \stackrel{3.A.9}{\leq} \frac{\beta\sqrt{n}\|\mathbf{f}(0) - \mathbf{y}\|_2}{\sqrt{m}\lambda_0} \leq R_v.$$

Similarly m ensures that

$$|g_k(t) - g_k(0)| \stackrel{3.A.10}{\leq} \frac{\beta^2 \sqrt{n} \|\mathbf{f}(0) - \mathbf{y}\|_2}{\sqrt{m} \lambda_0} \leq R_g.$$

The over-parametrization of m implies that the parameter trajectories stay close enough to initialization to satisfy the hypotheses of Lemmas 3.A.5, 3.A.6 and that $\lambda_{\min}(\mathbf{\Lambda}(t)) \geq \lambda_{\min}(\mathbf{V}(t))/\beta^2 \geq \frac{\lambda_0}{2\beta^2}$. To prove that $\lambda_{\min}(\mathbf{\Lambda}(t)) \geq \frac{\lambda_0}{2\beta^2}$ holds for all $0 \leq t \leq T$, we proceed by contradiction and suppose that one of Lemmas 3.A.9, 3.A.10 does not hold. Take T_0 to be the first failure time. Clearly $T_0 > 0$ and for $0 < t < T_0$ the above conditions hold, which implies that $\lambda_{\min}(\mathbf{V}(t)) \geq \frac{\lambda_0}{2}$ for $0 \leq t \leq T_0$; this contradicts one of Lemmas 3.A.9, 3.A.10 at time T_0 . Therefore we conclude that Lemmas 3.A.9, 3.A.10 hold for $t > 0$ and we can apply 3.A.8 to guarantee linear convergence. \square

Here we consider the case where the convergence is dominated by \mathbf{G} . This occurs when $\beta > 1$.

Proof of Theorem 3.A.2:

By substituting $m = \Omega(n^4 \log(n/\delta)/\beta^4 \mu_0^4)$ and using the bound on $\|\mathbf{f}(0) - \mathbf{y}\|_2$ of Lemma 3.A.11 we have that

$$\|\mathbf{v}_k(t) - \mathbf{v}_k(0)\|_2 \stackrel{3.A.9}{\leq} \frac{4\sqrt{n} \|\mathbf{f}(0) - \mathbf{y}\|_2}{\beta \mu_0 \sqrt{m}} \stackrel{3.A.11}{\leq} \frac{Cn\sqrt{\log(n/\delta)}}{\beta \mu_0 \sqrt{m}} \leq \tilde{R}_v.$$

Where the inequality is shown by a direct calculation substituting m .

This means that the parameter trajectories stay close enough to satisfy the hypotheses of Lemma 3.A.7 if $m = \Omega(n^4 \log(n/\delta)/\beta^4 \mu_0^4)$. Using the same argument as Theorem 3.A.1, we show that this holds for all $t > 0$. We proceed by contradiction, supposing that one of Lemmas 3.A.9, 3.A.10 do not hold. Take T_0 to be the first time one of the conditions of Lemmas 3.A.9, 3.A.10 fail. Clearly $T_0 > 0$ and for $0 < t < T_0$ the above derivation holds, which implies that $\lambda_{\min}(\mathbf{G}(t)) \geq \frac{\mu_0}{2}$. This contradicts Lemmas 3.A.9 3.A.10 at time T_0 , therefore we conclude that Lemma 3.A.8 holds for all $t > 0$ and guarantees linear convergence. \square

Note that if β is large, the required complexity on m is reduced. Taking $\beta = \Omega(\sqrt{n/\mu_0})$ gives the improved bound

$$m = \Omega\left(\frac{n^2 \log(n/\delta)}{\mu_0^2}\right).$$

3.A.2.2 Supporting lemmas and proofs of the lemmas used for gradient flow convergence

Proof of Lemma 3.5.1:

We prove Lemma 3.5.1 for \mathbf{V}^∞ , \mathbf{G}^∞ separately. \mathbf{V}^∞ can be viewed as the covariance matrix of the functionals ϕ_i defined as

$$\phi_i(\mathbf{v}) = \mathbf{x}_i \left(\mathbf{I} - \frac{\mathbf{v}\mathbf{v}^\top}{\|\mathbf{v}\|_2^2} \right) \mathbb{1}\{\mathbf{v}^\top \mathbf{x}_i \geq 0\} \quad (3.20)$$

over the Hilbert space \mathcal{V} of $L^2(N(0, \beta^2 \mathbf{I}))$ of functionals. Under this formulation, if $\phi_1, \phi_2, \dots, \phi_n$ are linearly independent, then \mathbf{V}^∞ is strictly positive definite. Thus, to show that \mathbf{V}^∞ is strictly positive definite is equivalent to showing that

$$c_1 \phi_1 + c_2 \phi_2 + \dots + c_n \phi_n = 0 \text{ in } \mathcal{V} \quad (3.21)$$

implies $c_i = 0$ for each i . The ϕ_i s are piece-wise continuous functionals, and equality in \mathcal{V} is equivalent to

$$c_1 \phi_1 + c_2 \phi_2 + \dots + c_n \phi_n = 0 \text{ almost everywhere.}$$

For the sake of contradiction, assume that there exist c_1, \dots, c_n that are not identically 0, satisfying (3.21). As c_i are not identically 0, there exists an i such that $c_i \neq 0$. We show this leads to a contradiction by constructing a non-zero measure region such that the linear combination $\sum_i c_i \phi_i$ is non-zero.

Denote the orthogonal subspace to \mathbf{x}_i as $D_i := \{\mathbf{v} \in \mathbb{R}^d : \mathbf{v}^\top \mathbf{x}_i = 0\}$. By Assumption 1,

$$D_i \not\subseteq \bigcup_{j \neq i} D_j$$

This holds since D_i is a $(d-1)$ -dimensional space which may not be written as the finite union of sub-spaces $D_i \cap D_j$ of dimension $d-2$ (since \mathbf{x}_i and \mathbf{x}_j are not parallel). Thus, take $\mathbf{z} \in D_i \setminus \bigcup_{j \neq i} D_j$. Since $\bigcup_{j \neq i} D_j$ is closed in \mathbb{R}^d , there exists an $R > 0$ such that

$$\mathcal{B}(\mathbf{z}, 4R) \cap \bigcup_{j \neq i} D_j = \emptyset.$$

Next take $\mathbf{y} \in \partial \mathcal{B}(\mathbf{z}, 3R) \cap D_i$ (where ∂ denotes the boundary) on the smaller disk of radius $3R$ so that it satisfies $\|\mathbf{y}\|_2 = \max_{\mathbf{y}' \in \partial \mathcal{B}(\mathbf{z}, 3R) \cap D_i} \|\mathbf{y}'\|_2$. Now for any $r \leq R$, the ball $\mathcal{B}(\mathbf{y}, r)$ is such that for all points $\mathbf{v} \in \mathcal{B}(\mathbf{y}, r)$ we have $\|\mathbf{v}^{\mathbf{x}_i^\perp}\|_2 \geq 2R$ and $\|\mathbf{v}^{\mathbf{x}_i}\|_2 \leq R$. Then for any $r \leq R$, the points $\mathbf{v} \in \mathcal{B}(\mathbf{y}, r) \subset \mathcal{B}(\mathbf{z}, 4R)$ satisfy that

$$\|\mathbf{x}_i^{\mathbf{v}^\perp}\|_2 \geq \|\mathbf{x}_i\|_2 - \frac{\mathbf{x}_i \cdot \mathbf{v}}{\|\mathbf{v}\|_2} \geq \|\mathbf{x}_i\|_2 \left(1 - \frac{R}{2R}\right) \geq \frac{\|\mathbf{x}_i\|_2}{2}.$$

Next we decompose the chosen ball $\mathcal{B}(\mathbf{y}, r) = B^+(r) \vee B^-(r)$ to the areas where the ReLU function at the point \mathbf{x}_i is active and inactive

$$B^+(r) = \mathcal{B}(\mathbf{y}, r) \cap \{\mathbf{x}_i^\top \mathbf{v} \geq 0\}, \quad B^-(r) = \mathcal{B}(\mathbf{y}, r) \cap \{\mathbf{x}_i^\top \mathbf{v} < 0\}.$$

Note that ϕ_i has a discontinuity on D_i and is continuous within each region $B^+(r)$ and $B^-(r)$. Moreover, for $j \neq i$, ϕ_j is continuous on the entire region of $\mathcal{B}(\mathbf{y}, r)$ since $\mathcal{B}(\mathbf{y}, r) \subset \mathcal{B}(\mathbf{z}, 4R) \subset D_j^c$. Since we have that ϕ_j is continuous in the region, the Lebesgue differentiation theorem implies that for $r \rightarrow 0$, ϕ_i satisfies on $B^+(r), B^-(r)$:

$$\lim_{r \rightarrow 0} \frac{1}{\mu(B^+(r))} \int_{B^+(r)} \phi_i = \mathbf{x}_i^{\mathbf{y}^\perp} \neq 0, \quad \lim_{r \rightarrow 0} \frac{1}{\mu(B^-(r))} \int_{B^-(r)} \phi_i = 0.$$

For $j \neq i$ ϕ_j is continuous on the entire ball $\mathcal{B}(\mathbf{y}, r)$ hence the Lebesgue differentiation theorem also gives

$$\lim_{r \rightarrow 0} \frac{1}{\mu(B^+(r))} \int_{B^+(r)} \phi_i = \phi_j(\mathbf{y}), \quad \lim_{r \rightarrow 0} \frac{1}{\mu(B^-(r))} \int_{B^-(r)} \phi_i = \phi_j(\mathbf{y}).$$

We integrate $c_1 \phi_1 + \dots + c_n \phi_n$ over $B^-(r)$ and $B^+(r)$ separately and subtract the integrals. By the assumption, $c_1 \phi_1 + \dots + c_n \phi_n = 0$ almost everywhere so each integral evaluates to 0 and the difference is also 0,

$$0 = \frac{1}{\mu(B^+(r))} \int_{B^+(r)} c_1 \phi_1 + \dots + c_n \phi_n - \frac{1}{\mu(B^-(r))} \int_{B^-(r)} c_1 \phi_1 + \dots + c_n \phi_n. \quad (3.22)$$

By the continuity of ϕ_j for $j \neq i$ taking $r \rightarrow 0$ we have that

$$\frac{1}{\mu(B^+(r))} \lim_{r \rightarrow 0} \int_{B^+(r)} \phi_j - \frac{1}{\mu(B^-(r))} \int_{B^-(r)} \phi_j = \phi_j(\mathbf{y}) - \phi_j(\mathbf{y}) = 0.$$

For ϕ_i the functionals evaluate differently. For $B^-(r)$ we have that

$$\frac{1}{\mu(B^-(r))} \lim_{r \rightarrow 0} \int_{B^-(r)} \phi_i = \frac{1}{\mu(B^-(r))} \lim_{r \rightarrow 0} \int_{B^-(r)} 0 = 0,$$

while the integral over the positive side, $B^+(r)$ is equal to

$$\frac{1}{\mu(B^+(r))} \int_{B^+(r)} \phi_i(\mathbf{z}) d\mathbf{z} = \frac{1}{\mu(B^+(r))} \int_{B^+(r)} \mathbf{x}_i^{\mathbf{z}^\perp} d\mathbf{z} = \mathbf{x}_i^{\mathbf{y}^\perp}.$$

By construction, $\|\mathbf{x}_i^{\mathbf{y}^\perp}\|_2 > R$ and is non-zero so we conclude that for (3.22) to hold we must have $c_i = 0$. This gives the desired contradiction and implies that ϕ_1, \dots, ϕ_n are independent and \mathbf{V}^∞ is positive definite with $\lambda_{\min}(\mathbf{V}^\infty) = \lambda_0$.

Next we consider \mathbf{G}^∞ and again frame the problem in the context of the covariance matrix of functionals. Define

$$\theta_i(\mathbf{v}) := \sigma \left(\frac{\mathbf{v}^\top \mathbf{x}_i}{\|\mathbf{v}\|_2} \right)$$

for $\mathbf{v} \neq 0$.

Now the statement of the theorem is equivalent to showing that the covariance matrix of $\{\theta_i\}$ does not have zero-eigenvalues, that is, the functionals θ_i s are linearly independent. For the sake of contradiction assume $\exists c_1, \dots, c_n$ such that

$$c_1\theta_1 + c_2\theta_2 + \dots + c_n\theta_n = 0 \text{ in } \mathcal{V} \text{ (equivalent to a.e.)}$$

Via the same contradiction argument we show that $c_i = 0$ for all i . Unlike ϕ_i defined in (3.20), each θ_i is continuous and non-negative so equality “a.e” is strengthened to “for all \mathbf{v} ”,

$$c_1\theta_1 + c_2\theta_2 + \dots + c_n\theta_n = 0.$$

Equality everywhere requires that the derivatives of the function are equal to 0 almost everywhere. Computing derivatives with respect to \mathbf{v} yields

$$c_1 \mathbf{x}_1^{\mathbf{v}^\perp} \mathbb{1}\{\mathbf{v}^\top \mathbf{x}_1 \geq 0\} + c_2 \mathbf{x}_2^{\mathbf{v}^\perp} \mathbb{1}\{\mathbf{v}^\top \mathbf{x}_2 \geq 0\} + \cdots + c_n \mathbf{x}_n^{\mathbf{v}^\perp} \mathbb{1}\{\mathbf{v}^\top \mathbf{x}_n \geq 0\} = 0.$$

Which coincide with

$$c_1 \phi_1 + \cdots + c_n \phi_n$$

By the first part of the proof, the linear combination $c_1 \phi_1 + \cdots + c_n \phi_n$ is non-zero around a ball of positive measure unless $c_i = 0$ for all i . This contradicts the assumption that the derivative is 0 almost everywhere; therefore \mathbf{G}^∞ is strictly positive definite with $\lambda_{\min}(\mathbf{G}^\infty) =: \mu_0 > 0$. \square

We briefly derive an inequality for the sum of indicator functions for events that are bounded by the sum of indicator functions of *independent* events. This enables us to develop more refined concentration than in [DZP19] for monitoring the orthogonal and aligned Gram matrices during training.

Lemma 3.A.13. *Let A_1, \dots, A_m be a sequence of events and suppose that $A_k \subseteq B_k$ with B_1, \dots, B_m mutually independent. Further assume that for each k , $\mathbb{P}(B_k) \leq p$, and define $S = \frac{1}{m} \sum_{k=1}^m \mathbb{1}_{A_k}$. Then with probability $1 - \delta$, S satisfies*

$$S \leq p \left(2 + \frac{8 \log(1/\delta)}{3mp} \right).$$

Proof of Lemma 3.A.13:

Bound S as

$$S = \frac{1}{m} \sum_{k=1}^m \mathbb{1}_{A_k} \leq \frac{1}{m} \sum_{k=1}^m \mathbb{1}_{B_k}.$$

We apply Bernstein's concentration inequality to reach the bound. Denote $X_k = \frac{\mathbb{1}_{B_k}}{m}$ and $\tilde{S} = \sum_{k=1}^m X_k$. Then

$$\text{Var}(X_k) \leq \mathbb{E}X_k^2 = (1/m)^2 \mathbb{P}(X_k) + 0 \leq \frac{p}{m^2}, \quad \mathbb{E}\tilde{S} = \mathbb{E} \sum_{k=1}^m X_k \leq p.$$

Applying Bernstein's inequality yields

$$\mathbb{P}(\tilde{S} - \mathbb{E}\tilde{S} \geq t) \leq \exp\left(\frac{-t^2/2}{\sum_{k=1}^m \mathbb{E}X_k^2 + \frac{t}{3m}}\right).$$

Fix δ and take the smallest t such that $\mathbb{P}(\tilde{S} - \mathbb{E}\tilde{S} \geq t) \leq \delta$. Denote $t = r \cdot \mathbb{E}\tilde{S}$, either $\mathbb{P}(\tilde{S} - \mathbb{E}\tilde{S} \geq \mathbb{E}\tilde{S}) \leq \delta$, or $t = r\mathbb{E}\tilde{S}$ corresponds to $r \geq 1$. Note that $t = r\mathbb{E}\tilde{S} \leq rp$. In the latter case, the bound is written as

$$\mathbb{P}(\tilde{S} - \mathbb{E}\tilde{S} \geq rp) \leq \exp\left(\frac{-(pr)^2/2}{p/m + \frac{pr}{3m}}\right) \leq \exp\left(\frac{-(pr)^2/2}{\frac{p}{m}(1 + \frac{r}{3})}\right) \leq \exp\left(\frac{-(pr)^2/2}{\frac{p}{m}(\frac{4r}{3})}\right) = \exp\left(\frac{-3prm}{8}\right).$$

Solving for δ gives

$$rp \leq \frac{8 \log(1/\delta)}{3m}.$$

Hence with probability $1 - \delta$,

$$S \leq \tilde{S} \leq \max\left\{p\left(1 + \frac{8 \log(1/\delta)}{3mp}\right), 2p\right\} \leq p\left(2 + \frac{8 \log(1/\delta)}{3mp}\right).$$

□

Proof of Lemma 3.A.3:

We prove the claim by applying concentration on each entry of the difference matrix. Each entry $\mathbf{V}_{ij}(0)$ is written as

$$\mathbf{V}_{ij}(0) = \frac{1}{m} \sum_{k=1}^m \langle \mathbf{x}_i^{\mathbf{v}_k(0)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(0)^\perp} \rangle \left(\frac{\beta c_k \cdot g_k}{\|\mathbf{v}_k\|_2} \right)^2 \mathbf{1}_{ik}(0) \mathbf{1}_{jk}(0).$$

At initialization $g_k(0) = \|\mathbf{v}_k(0)\|_2/\beta$, $c_k^2 = 1$ so $\mathbf{V}_{ij}(0)$ simplifies to

$$\mathbf{V}_{ij}(0) = \frac{1}{m} \sum_{k=1}^m \langle \mathbf{x}_i^{\mathbf{v}_k(0)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(0)^\perp} \rangle \mathbf{1}_{ik}(0) \mathbf{1}_{jk}(0).$$

Since the weights $\mathbf{v}_k(0)$ are initialized independently for each entry we have $\mathbb{E}_{\mathbf{v}} \mathbf{V}_{ij}(0) = \mathbf{V}_{ij}^\infty$. We measure the deviation $\mathbf{V}(0) - \mathbf{V}^\infty$ via concentration. Each term in the sum $\frac{1}{m} \sum_{j=1}^m \langle \mathbf{x}_i^{\mathbf{v}_k(0)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(0)^\perp} \rangle \mathbf{1}_{ik}(0) \mathbf{1}_{jk}(0)$ is independent and bounded,

$$-1 \leq \langle \mathbf{x}_i^{\mathbf{v}_k(0)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(0)^\perp} \rangle \mathbf{1}_{ik}(0) \mathbf{1}_{jk}(0) \leq 1.$$

Applying Hoeffding's inequality to each entry yields that with probability $1 - \delta/n^2$, for all i, j ,

$$|\mathbf{V}_{ij}(0) - \mathbf{V}_{ij}^\infty| \leq \frac{2\sqrt{\log(n^2/\delta)}}{\sqrt{m}}.$$

Taking a union bound over all entries, with probability $1 - \delta$,

$$|\mathbf{V}_{ij}(0) - \mathbf{V}_{ij}^\infty| \leq \frac{4\sqrt{\log(n/\delta)}}{\sqrt{m}}.$$

Bounding the spectral norm, with probability $1 - \delta$,

$$\begin{aligned} \|\mathbf{V}(0) - \mathbf{V}^\infty\|_2^2 &\leq \|\mathbf{V}(0) - \mathbf{V}^\infty\|_F^2 \leq \sum_{i,j} |\mathbf{V}_{ij}(0) - \mathbf{V}_{ij}^\infty|^2 \\ &\leq \frac{16n^2 \log(n/\delta)}{m}. \end{aligned}$$

Taking $m = \Omega\left(\frac{n^2 \log(n/\delta)}{\lambda_0^2}\right)$ therefore guarantees

$$\|\mathbf{V}(0) - \mathbf{V}^\infty\|_2 \leq \frac{\lambda_0}{4}.$$

□

Proof of Lemma 3.A.4:

This is completely analogous to 3.A.3. Recall $\mathbf{G}(0)$ is defined as,

$$\mathbf{G}_{ij}(0) = \frac{1}{m} \sum_{k=1}^m \langle \mathbf{x}_i^{\mathbf{v}_k(0)}, \mathbf{x}_j^{\mathbf{v}_k(0)} \rangle c_k^2 \mathbb{1}_{ik}(0) \mathbb{1}_{jk}(0)$$

with $c_k^2 = 1$ and $\mathbf{v}_k(0) \sim N(0, \beta^2 \mathbf{I})$ are initialized i.i.d. Since each term is bounded like 3.A.3.

The same analysis gives

$$\|\mathbf{G}_{ij}(0) - \mathbf{G}_{ij}^\infty\|_2^2 \leq \frac{16n^2 \log(n/\delta)}{m}.$$

Taking $m = \Omega\left(\frac{n^2 \log(n/\delta)}{\mu_0^2}\right)$ therefore guarantees,

$$\|\mathbf{G}(0) - \mathbf{G}^\infty\|_2 \leq \frac{\mu_0}{4}.$$

□

Proof of Lemma 3.A.5:

For a given R , define the event of a possible sign change of neuron k at point \mathbf{x}_i as

$$A_{i,k}(R) = \{\exists \mathbf{v} : \|\mathbf{v} - \mathbf{v}_k(0)\|_2 \leq R, \text{ and } \mathbb{1}\{\mathbf{v}_k(0)^\top \mathbf{x}_i \geq 0\} \neq \mathbb{1}\{\mathbf{v}^\top \mathbf{x}_i \geq 0\}\}$$

$A_{i,k}(R)$ occurs exactly when $|\mathbf{v}_k(0)^\top \mathbf{x}_i| \leq R$, since $\|\mathbf{x}_i\|_2 = 1$ and the perturbation may be taken in the direction of $-\mathbf{x}_i$. To bound the probability $A_{i,k}(R)$ we consider the probability of the event

$$\mathbb{P}(A_{i,k}(R)) = \mathbb{P}(|\mathbf{v}_k(0)^\top \mathbf{x}_i| < R) = \mathbb{P}(|z| < R).$$

Here, $z \sim N(0, \beta^2)$ since the product $\mathbf{v}_k(0)^\top \mathbf{x}_i$ follows a centered normal distribution. The norm of $\|\mathbf{x}_i\|_2 = 1$ which implies that z computes to a standard deviation β . Via estimates on the normal distribution, the probability on the event is bounded like

$$\mathbb{P}(A_{i,k}(R)) \leq \frac{2R}{\beta\sqrt{2\pi}}.$$

We use the estimate for $\mathbb{P}(A_{i,k}(R))$ to bound the difference between the surrogate Gram matrix and the Gram matrix at initialization $\mathbf{V}(0)$.

Recall the surrogate $\hat{\mathbf{V}}(t)$ is defined as

$$\hat{\mathbf{V}}_{ij}(t) = \frac{1}{m} \sum_{k=1}^m \langle \mathbf{x}_i^{\mathbf{v}_k(t)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(t)^\perp} \rangle \mathbb{1}_{ik}(t) \mathbb{1}_{jk}(t).$$

Thus for entry i, j we have

$$|\hat{\mathbf{V}}_{ij}(t) - \mathbf{V}_{ij}(0)| = \left| \frac{1}{m} \sum_{k=1}^m \langle \mathbf{x}_i^{\mathbf{v}_k(t)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(t)^\perp} \rangle \mathbb{1}_{ik}(t) \mathbb{1}_{jk}(t) - \langle \mathbf{x}_i^{\mathbf{v}_k(0)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(0)^\perp} \rangle \mathbb{1}_{ik}(0) \mathbb{1}_{jk}(0) \right|$$

This sum is decomposed into the difference between the inner product and the difference in the rectifier patterns terms respectively:

$$\left(\langle \mathbf{x}_i^{\mathbf{v}_k(t)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(t)^\perp} \rangle - \langle \mathbf{x}_i^{\mathbf{v}_k(0)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(0)^\perp} \rangle \right), \quad \left(\mathbb{1}_{ik}(t) \mathbb{1}_{jk}(t) - \mathbb{1}_{ik}(0) \mathbb{1}_{jk}(0) \right).$$

Define

$$Y_{ij}^k = \left(\langle \mathbf{x}_i^{\mathbf{v}_k(t)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(t)^\perp} \rangle - \langle \mathbf{x}_i^{\mathbf{v}_k(0)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(0)^\perp} \rangle \right) (\mathbb{1}_{ik}(t) \mathbb{1}_{jk}(t)),$$

$$Z_{ij}^k = \left(\langle \mathbf{x}_i^{\mathbf{v}_k(0)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(0)^\perp} \rangle \right) \left(\mathbb{1}_{ik}(t) \mathbb{1}_{jk}(t) - \mathbb{1}_{ik}(0) \mathbb{1}_{jk}(0) \right).$$

Then

$$|\hat{\mathbf{V}}_{ij}(t) - \mathbf{V}_{ij}(0)| = \left| \frac{1}{m} \sum_{k=1}^m Y_{ij}^k + Z_{ij}^k \right| \leq \left| \frac{1}{m} \sum_{k=1}^m Y_{ij}^k \right| + \left| \frac{1}{m} \sum_{k=1}^m Z_{ij}^k \right|.$$

To bound $|\frac{1}{m} \sum_{k=1}^m Y_{ij}^k|$ we bound each $|Y_{ij}^k|$ as follows.

$$\begin{aligned} |Y_{ij}^k| &= \left| \left(\langle \mathbf{x}_i^{\mathbf{v}_k(t)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(t)^\perp} \rangle - \langle \mathbf{x}_i^{\mathbf{v}_k(0)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(0)^\perp} \rangle \right) (\mathbb{1}_{ik}(t) \mathbb{1}_{jk}(t)) \right| \\ &\leq \left| \langle \mathbf{x}_i^{\mathbf{v}_k(t)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(t)^\perp} \rangle - \langle \mathbf{x}_i^{\mathbf{v}_k(0)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(0)^\perp} \rangle \right| \\ &= \left| \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \langle \mathbf{x}_i^{\mathbf{v}_k(t)}, \mathbf{x}_j^{\mathbf{v}_k(t)} \rangle + \langle \mathbf{x}_i^{\mathbf{v}_k(0)}, \mathbf{x}_j^{\mathbf{v}_k(0)} \rangle - \langle \mathbf{x}_i, \mathbf{x}_j \rangle \right| \\ &= \left| \left\langle \frac{\mathbf{x}_i^\top \mathbf{v}_k(t)}{\|\mathbf{v}_k(t)\|_2} \cdot \frac{\mathbf{v}_k(t)}{\|\mathbf{v}_k(t)\|_2}, \frac{\mathbf{x}_j^\top \mathbf{v}_k(t)}{\|\mathbf{v}_k(t)\|_2} \cdot \frac{\mathbf{v}_k(t)}{\|\mathbf{v}_k(t)\|_2} \right\rangle - \langle \mathbf{x}_i^{\mathbf{v}_k(0)}, \mathbf{x}_j^{\mathbf{v}_k(0)} \rangle \right| \\ &= \left| \frac{\mathbf{x}_i^\top \mathbf{v}_k(t)}{\|\mathbf{v}_k(t)\|_2} \cdot \frac{\mathbf{x}_j^\top \mathbf{v}_k(t)}{\|\mathbf{v}_k(t)\|_2} - \langle \mathbf{x}_i^{\mathbf{v}_k(0)}, \mathbf{x}_j^{\mathbf{v}_k(0)} \rangle \right| \\ &= \left| \frac{\mathbf{x}_i^\top \mathbf{v}_k(0)}{\|\mathbf{v}_k(0)\|_2} \cdot \frac{\mathbf{x}_j^\top \mathbf{v}_k(0)}{\|\mathbf{v}_k(0)\|_2} + \mathbf{x}_i^\top \left(\frac{\mathbf{v}_k(t)}{\|\mathbf{v}_k(t)\|_2} - \frac{\mathbf{v}_k(0)}{\|\mathbf{v}_k(0)\|_2} \right) \cdot \frac{\mathbf{x}_j^\top \mathbf{v}_k(t)}{\|\mathbf{v}_k(t)\|_2} \right. \\ &\quad \left. + \mathbf{x}_j^\top \left(\frac{\mathbf{v}_k(t)}{\|\mathbf{v}_k(t)\|_2} - \frac{\mathbf{v}_k(0)}{\|\mathbf{v}_k(0)\|_2} \right) \cdot \frac{\mathbf{x}_i^\top \mathbf{v}_k(0)}{\|\mathbf{v}_k(0)\|_2} - \langle \mathbf{x}_i^{\mathbf{v}_k(0)}, \mathbf{x}_j^{\mathbf{v}_k(0)} \rangle \right| \\ &\leq \left| \mathbf{x}_i^\top \left(\frac{\mathbf{v}_k(t)}{\|\mathbf{v}_k(t)\|_2} - \frac{\mathbf{v}_k(0)}{\|\mathbf{v}_k(0)\|_2} \right) \cdot \frac{\mathbf{x}_j^\top \mathbf{v}_k(t)}{\|\mathbf{v}_k(t)\|_2} \right| + \left| \mathbf{x}_i^\top \left(\frac{\mathbf{v}_k(t)}{\|\mathbf{v}_k(t)\|_2} - \frac{\mathbf{v}_k(0)}{\|\mathbf{v}_k(0)\|_2} \right) \cdot \frac{\mathbf{x}_j^\top \mathbf{v}_k(t)}{\|\mathbf{v}_k(t)\|_2} \right| \\ &\leq 2 \left\| \frac{\mathbf{v}_k(t)}{\|\mathbf{v}_k(t)\|_2} - \frac{\mathbf{v}_k(0)}{\|\mathbf{v}_k(0)\|_2} \right\|_2. \end{aligned}$$

Therefore, we have

$$\begin{aligned} \left| \frac{1}{m} \sum_{k=1}^m Y_{ij}^k \right| &\leq \frac{2}{m} \sum_{k=1}^m \left\| \frac{\mathbf{v}_k(t)}{\|\mathbf{v}_k(t)\|_2} - \frac{\mathbf{v}_k(0)}{\|\mathbf{v}_k(0)\|_2} \right\|_2 \\ &\leq \frac{4R_{\mathbf{v}}(2m/\delta)^{1/d}}{\beta} \\ &\leq \frac{8R_{\mathbf{v}}(m/\delta)^{1/d}}{\beta}, \end{aligned}$$

where the first inequality follows from Lemma 3.A.12. Note that the inequality holds with high probability $1 - \delta/2$ for all i, j .

For the second sum, $|\frac{1}{m} \sum_{k=1}^m Z_{ij}^k| \leq \frac{1}{m} \sum_{k=1}^m \mathbb{1}_{A_{ik}(R)} + \frac{1}{m} \sum_{k=1}^m \mathbb{1}_{A_{jk}(R)}$ so we apply Lemma 3.A.13 to get, with probability $1 - \delta/2n^2$

$$\begin{aligned} \left| \frac{1}{m} \sum_{k=1}^m Z_{ij}^k \right| &\leq \frac{2R_v}{\beta\sqrt{2\pi}} \left(2 + \frac{2\sqrt{2\pi}\beta \log(2n^2/\delta)}{3mR_v} \right) \\ &\leq \frac{8R_v}{\beta\sqrt{2\pi}}, \end{aligned}$$

since m satisfies $m = \Omega\left(\frac{(m/\delta)^{1/d} n^2 \log(n/\delta)}{\lambda_0}\right)$. Combining the two sums for Y_{ij}^k and Z_{ij}^k , with probability $1 - \frac{\delta}{2n^2}$,

$$|\hat{\mathbf{V}}_{ij}(t) - \mathbf{V}_{ij}(0)| \leq \frac{8R_v}{\beta\sqrt{2\pi}} + \frac{8R_v(m/\delta)^{1/d}}{\beta} \leq \frac{12R_v(m/\delta)^{1/d}}{\beta}.$$

Taking a union bound, with probability $1 - \delta/2$,

$$\|\hat{\mathbf{V}}(t) - \mathbf{V}(0)\|_F = \sqrt{\sum_{i,j} |\hat{\mathbf{V}}_{ij}(t) - \mathbf{V}_{ij}(0)|^2} \leq \frac{12nR_v(m/\delta)^{1/d}}{\beta}.$$

Bounding the spectral norm by the Frobenous norm,

$$\|\hat{\mathbf{V}}(t) - \mathbf{V}(0)\|_2 \leq \frac{12nR_v(m/\delta)^{1/d}}{\beta}.$$

Taking $R_v = \frac{\beta\lambda_0}{96n(m/\delta)^{1/d}}$ gives the desired bound.

$$\|\hat{\mathbf{V}}(t) - \mathbf{V}(0)\|_2 \leq \frac{\lambda_0}{8}.$$

□

Proof of Lemma 3.A.6:

To bound $\|\mathbf{V}(t) - \mathbf{V}(0)\|_2$ we now consider $\|\mathbf{V}(t) - \hat{\mathbf{V}}(t)\|_2$. The entries of $\mathbf{V}_{ij}(t)$ are given as

$$\mathbf{V}_{ij}(t) = \frac{1}{m} \sum_{k=1}^m \langle \mathbf{x}_i^{\mathbf{v}_k(t)^\perp}, x_j^{\mathbf{v}_k(t)^\perp} \rangle \mathbb{1}_{ik}(t) \mathbb{1}_{jk}(t) \left(\frac{\beta c_k \cdot g_k}{\|\mathbf{v}_k(0)\|_2} \right)^2.$$

The surrogate $\hat{\mathbf{V}}(t)$ is defined as

$$\hat{\mathbf{V}}_{ij}(t) = \frac{1}{m} \sum_{k=1}^m \langle \mathbf{x}_i^{\mathbf{v}_k(t)^\perp}, x_j^{\mathbf{v}_k(t)^\perp} \rangle \mathbb{1}_{ik}(t) \mathbb{1}_{jk}(t).$$

The only difference is in the second layer terms. The difference between each entry is written as

$$\begin{aligned} |\mathbf{V}_{ij}(t) - \hat{\mathbf{V}}_{ij}(t)| &= \left| \frac{1}{m} \sum_{k=1}^m \langle \mathbf{x}_i^{\mathbf{v}_k(t)^\perp}, x_j^{\mathbf{v}_k(t)^\perp} \rangle \mathbb{1}_{ik}(t) \mathbb{1}_{jk}(t) \left(\left(\frac{\beta c_k \cdot g_k}{\|\mathbf{v}_k(t)\|_2} \right)^2 - 1 \right) \right| \\ &\leq \max_{1 \leq k \leq m} \left(\frac{\beta^2 g_k(t)^2}{\|\mathbf{v}_k(t)\|_2^2} - 1 \right). \end{aligned}$$

Write $1 = \frac{\beta^2 g_k^2(0)}{\|\mathbf{v}_k(0)\|_2^2}$, since $\|\mathbf{v}_k(t)\|_2$ is increasing in t according to (3.5)

$$\frac{\beta^2 g_k(t)^2}{\|\mathbf{v}_k(t)\|_2^2} - 1 = \frac{\beta^2 g_k(t)^2}{\|\mathbf{v}_k(t)\|_2^2} - \frac{\beta^2 g_k(0)^2}{\|\mathbf{v}_k(0)\|_2^2} \leq 3R_g(m/\delta)^{1/d} + 3R_v(m/\delta)^{1/d}/\beta.$$

The above inequality is shown by considering different cases for the sign of the difference $g_k(t) - g_k(0)$. Now

$$\begin{aligned} \left| \frac{\beta^2 g_k(t)^2}{\|\mathbf{v}_k(t)\|_2^2} - \frac{\beta^2 g_k(0)^2}{\|\mathbf{v}_k(0)\|_2^2} \right| &= \left| \left(\frac{\beta g_k(t)}{\|\mathbf{v}_k(t)\|_2} + \frac{\beta g_k(0)}{\|\mathbf{v}_k(0)\|_2} \right) \left(\frac{\beta g_k(t)}{\|\mathbf{v}_k(t)\|_2} - \frac{\beta g_k(0)}{\|\mathbf{v}_k(0)\|_2} \right) \right| \\ &\leq \left| \left(\frac{\beta g_k(0) + \beta R_g}{\|\mathbf{v}_k(0)\|_2} + \frac{\beta g_k(0)}{\|\mathbf{v}_k(0)\|_2} \right) \left(\frac{\beta g_k(t)}{\|\mathbf{v}_k(t)\|_2} - \frac{\beta g_k(0)}{\|\mathbf{v}_k(0)\|_2} \right) \right| \\ &\leq (2 + R_g(m/\delta)^{1/d}) \left| \left(\frac{\beta g_k(t)}{\|\mathbf{v}_k(t)\|_2} - \frac{\beta g_k(0)}{\|\mathbf{v}_k(0)\|_2} \right) \right| \\ &\leq (2 + R_g(m/\delta)^{1/d}) \max \left(\left| \frac{\beta(g_k(0) + R_g)}{\|\mathbf{v}_k(0)\|_2} - \frac{\beta g_k(0)}{\|\mathbf{v}_k(0)\|_2} \right|, \left| \frac{\beta(g_k(0) - R_g)}{\|\mathbf{v}_k(0)\|_2 + R_v} - \frac{\beta g_k(0)}{\|\mathbf{v}_k(0)\|_2} \right| \right) \\ &\leq (2 + R_g(m/\delta)^{1/d}) \max (R_g(m/\delta)^{1/d}, R_g(m/\delta)^{1/d} + R_v(m/\delta)^{1/d}/\beta) \\ &\leq 3R_g(m/\delta)^{1/d} + 3R_v(m/\delta)^{1/d}/\beta, \end{aligned}$$

where the second inequality holds due to Lemma 3.A.12 with probability $1 - \delta$ over the initialization.

Hence:

$$\begin{aligned} \|\hat{\mathbf{V}}(t) - \mathbf{V}(t)\|_2 &\leq \|\hat{\mathbf{V}}(t) - \mathbf{V}(t)\|_F = \sqrt{\sum_{i,j} |\hat{\mathbf{V}}_{ij}(t) - \mathbf{V}_{ij}(t)|^2} \\ &\leq 3nR_g(m/\delta)^{1/d} + 3nR_v(m/\delta)^{1/d}/\beta. \end{aligned}$$

Substituting R_v, R_g gives

$$\|\hat{\mathbf{V}}(t) - \mathbf{V}(t)\|_2 \leq \frac{\lambda_0}{8}.$$

Now we use Lemma 3.A.5 to get that with probability $1 - \delta$

$$\|\hat{\mathbf{V}}(t) - \mathbf{V}(0)\|_2 \leq \frac{\lambda_0}{8}.$$

Combining, we get with probability $1 - \delta$

$$\|\mathbf{V}(t) - \mathbf{V}(0)\|_2 \leq \frac{\lambda_0}{4}.$$

We note that the source for all the high probability uncertainty $1 - \delta$ all arise from initialization and the application of Lemma 3.A.12. \square

Proof of Lemma 3.A.7:

To prove the claim we consider each entry i, j of $\mathbf{G}(t) - \mathbf{G}(0)$. We have,

$$\begin{aligned} |\mathbf{G}_{ij}(t) - \mathbf{G}_{ij}(0)| &= \left| \frac{1}{m} \sum_{k=1}^m \sigma\left(\frac{\mathbf{v}_k(t)^\top \mathbf{x}_i}{\|\mathbf{v}_k(t)\|_2}\right) \sigma\left(\frac{\mathbf{v}_k(t)^\top \mathbf{x}_j}{\|\mathbf{v}_k(t)\|_2}\right) - \sigma\left(\frac{\mathbf{v}_k(0)^\top \mathbf{x}_i}{\|\mathbf{v}_k(0)\|_2}\right) \sigma\left(\frac{\mathbf{v}_k(0)^\top \mathbf{x}_j}{\|\mathbf{v}_k(0)\|_2}\right) \right| \\ &\leq \frac{1}{m} \left| \sum_{k=1}^m \sigma\left(\frac{\mathbf{v}_k(t)^\top \mathbf{x}_i}{\|\mathbf{v}_k(t)\|_2}\right) \sigma\left(\frac{\mathbf{v}_k(t)^\top \mathbf{x}_j}{\|\mathbf{v}_k(t)\|_2}\right) - \sigma\left(\frac{\mathbf{v}_k(t)^\top \mathbf{x}_i}{\|\mathbf{v}_k(t)\|_2}\right) \sigma\left(\frac{\mathbf{v}_k(0)^\top \mathbf{x}_j}{\|\mathbf{v}_k(0)\|_2}\right) \right| \\ &\quad + \frac{1}{m} \left| \sum_{k=1}^m \sigma\left(\frac{\mathbf{v}_k(t)^\top \mathbf{x}_i}{\|\mathbf{v}_k(t)\|_2}\right) \sigma\left(\frac{\mathbf{v}_k(0)^\top \mathbf{x}_j}{\|\mathbf{v}_k(0)\|_2}\right) - \sigma\left(\frac{\mathbf{v}_k(0)^\top \mathbf{x}_i}{\|\mathbf{v}_k(0)\|_2}\right) \sigma\left(\frac{\mathbf{v}_k(0)^\top \mathbf{x}_j}{\|\mathbf{v}_k(0)\|_2}\right) \right| \\ &\leq 2 \left\| \frac{\mathbf{v}_k(t)}{\|\mathbf{v}_k(t)\|_2} - \frac{\mathbf{v}_k(0)}{\|\mathbf{v}_k(0)\|_2} \right\|_2 \leq \frac{2\tilde{R}_v(m/\delta)^{1/d}}{\beta}. \end{aligned}$$

In the last inequality we used the fact that

$$\left\| \frac{\mathbf{v}_k(0)}{\|\mathbf{v}_k(0)\|_2} - \frac{\mathbf{v}_k(t)}{\|\mathbf{v}_k(t)\|_2} \right\|_2 \leq \frac{\|\mathbf{v}_k(t) - \mathbf{v}_k(0)\|_2}{\|\mathbf{v}_k(0)\|_2} \leq \frac{(m/\delta)^{1/d}}{\beta} \|\mathbf{v}_k(t) - \mathbf{v}_k(0)\|_2,$$

where the first inequality uses that $\|\mathbf{v}_k(0)\|_2 \leq \|\mathbf{v}_k(t)\|_2$ and is intuitive from a geometrical

standpoint. Algebraically given vectors \mathbf{a}, \mathbf{b} , then for any $c \geq 1$

$$\begin{aligned} \left\| \frac{\mathbf{a}c}{\|\mathbf{a}\|_2} - \frac{\mathbf{b}}{\|\mathbf{b}\|_2} \right\|_2^2 &= \left\| \frac{\mathbf{a}}{\|\mathbf{a}\|_2} - \frac{\mathbf{b}}{\|\mathbf{b}\|_2} + (c-1) \frac{\mathbf{a}}{\|\mathbf{a}\|_2} \right\|_2^2 \\ &= \left\| \frac{\mathbf{a}}{\|\mathbf{a}\|_2} - \frac{\mathbf{b}}{\|\mathbf{b}\|_2} \right\|_2^2 + (c-1)^2 + 2(c-1) \left\langle \frac{\mathbf{a}}{\|\mathbf{a}\|_2} - \frac{\mathbf{b}}{\|\mathbf{b}\|_2}, \frac{\mathbf{a}}{\|\mathbf{a}\|_2} \right\rangle \\ &\geq \left\| \frac{\mathbf{a}}{\|\mathbf{a}\|_2} - \frac{\mathbf{b}}{\|\mathbf{b}\|_2} \right\|_2^2 + (c-1)^2 \geq \left\| \frac{\mathbf{a}}{\|\mathbf{a}\|_2} - \frac{\mathbf{b}}{\|\mathbf{b}\|_2} \right\|_2^2. \end{aligned}$$

The first inequality in the line above is since $\frac{\langle \mathbf{a}, \mathbf{b} \rangle}{\|\mathbf{a}\|_2 \|\mathbf{b}\|_2} \leq 1$.

Hence,

$$\|\mathbf{G}(t) - \mathbf{G}(0)\|_2 \leq \|\mathbf{G}(t) - \mathbf{G}(0)\|_F = \sqrt{\sum_{i,j} |\mathbf{G}_{ij}(t) - \mathbf{G}_{ij}(0)|^2} \leq \frac{2n\tilde{R}_v(m/\delta)^{1/d}}{\beta\sqrt{2\pi}}.$$

Taking $\tilde{R}_v = \frac{\sqrt{2\pi}\beta\mu_0}{8n(m/\delta)^{1/d}}$ gives the desired bound. Therefore, with probability $1 - \delta$,

$$\|\mathbf{G}(t) - \mathbf{G}(0)\|_2 \leq \frac{\mu_0}{4}.$$

□

Now that we have established bounds on $\mathbf{V}(t), \mathbf{G}(t)$ given that the parameters stay near initialization, we show that the evolution converges in that case:

Proof of Lemma 3.A.8:

Consider the squared norm of the predictions $\|\mathbf{f}(t) - \mathbf{y}\|_2^2$. Taking the derivative of the loss with respect to time,

$$\frac{d}{dt} \|\mathbf{f}(t) - \mathbf{y}\|_2^2 = -2(\mathbf{f}(t) - \mathbf{y})^\top \left(\mathbf{G}(t) + \frac{\mathbf{V}(t)}{\beta^2} \right) (\mathbf{f}(t) - \mathbf{y}).$$

Since we assume that $\lambda_{\min} \left(\mathbf{G}(t) + \frac{\mathbf{V}(t)}{\beta^2} \right) \geq \frac{\omega}{2}$, the derivative of the squared norm is bounded as

$$\frac{d}{dt} \|\mathbf{f}(t) - \mathbf{y}\|_2^2 \leq -\omega \|\mathbf{f}(t) - \mathbf{y}\|_2^2.$$

Applying an integrating factor yields

$$\|\mathbf{f}(t) - \mathbf{y}\|_2^2 \exp(\omega t) \leq C.$$

Substituting the initial conditions, we get

$$\|\mathbf{f}(t) - \mathbf{y}\|_2^2 \leq \exp(-\omega t) \|\mathbf{f}(0) - \mathbf{y}\|_2^2.$$

□

For now, assuming the linear convergence derived in Lemma 3.A.8, we bound the distance of the parameters from initialization. Later we combine the bound on the parameters and Lemmas 3.A.6, 3.A.7 bounding the least eigenvalue of $\Lambda(t)$, to derive a condition on the over-parametrization m and ensure convergence from random initialization.

Proof of Lemma 3.A.9:

Denote $f(\mathbf{x}_i)$ at time t as $f_i(t)$. Since $\|\mathbf{x}_i^{\mathbf{v}_k(t)^\perp}\|_2 \leq \|\mathbf{x}_i\|_2 = 1$, we have that

$$\begin{aligned} \left\| \frac{d\mathbf{v}_k(t)}{dt} \right\|_2 &= \left\| \sum_{i=1}^n (y_i - f_i(t)) \frac{1}{\sqrt{m}} c_k g_k(t) \frac{1}{\|\mathbf{v}_k(t)\|_2} \mathbf{x}_i^{\mathbf{v}_k(t)^\perp} \mathbf{1}_{ik}(t) \right\|_2 \\ &\leq \frac{1}{\sqrt{m}} \sum_{i=1}^n |y_i - f_i(t)| \frac{c_k g_k(t)}{\|\mathbf{v}_k(t)\|_2}. \end{aligned}$$

Now using (3.5) and the initialization $\|\mathbf{v}_k(0)\| = \beta g_k(0)$, we bound $\left| \frac{c_k g_k(t)}{\|\mathbf{v}_k(t)\|_2} \right|$,

$$\left| \frac{c_k g_k(t)}{\|\mathbf{v}_k(t)\|_2} \right| \leq \left| c_k \left(\frac{g_k(0) + R_g}{\|\mathbf{v}_k(0)\|_2} \right) \right| \leq \frac{1}{\beta} \left(1 + \beta R_g / \|\mathbf{v}_k(0)\|_2 \right).$$

By Lemma 3.A.12, we have that with probability $1 - \delta$ over the initialization,

$$\beta / \|\mathbf{v}_k(0)\|_2 \leq C(m/\delta)^{1/d}.$$

Hence $\beta R_g / \|\mathbf{v}_k(0)\|_2 \leq 1$. This fact bounds $\left| \frac{c_k g_k(t)}{\|\mathbf{v}_k(t)\|_2} \right|$ with probability $1 - \delta$ for each k ,

$$\left| \frac{c_k g_k(t)}{\|\mathbf{v}_k(t)\|_2} \right| \leq 2/\beta.$$

Substituting the bound,

$$\begin{aligned} \left\| \frac{d}{dt} \mathbf{v}_k(t) \right\|_2 &\leq \frac{2}{\beta\sqrt{m}} \sum_{i=1}^n |f_i(t) - y_i| \\ &\leq \frac{2\sqrt{n}}{\beta\sqrt{m}} \|\mathbf{f}(t) - \mathbf{y}\|_2 \\ &\leq \frac{2\sqrt{n}}{\beta\sqrt{m}} \exp(-\omega t/2) \|\mathbf{f}(0) - \mathbf{y}\|_2. \end{aligned}$$

Thus, integrating and applying Jensen's inequality,

$$\|\mathbf{v}_k(t) - \mathbf{v}_k(0)\|_2 \leq \int_0^t \left\| \frac{d\mathbf{v}_k(s)}{ds} \right\|_2 ds \leq \frac{4\sqrt{n} \|\mathbf{f}(0) - \mathbf{y}\|_2}{\beta\omega\sqrt{m}}.$$

Note that the condition $|g_k(t) - g_k(0)| \leq R_g$ is stronger than needed and merely assuring that $|g_k(t) - g_k(0)| \leq 1/(m/\delta)^{1/d}$ suffices. \square

Analogously we derive bounds for the distance of g_k from initialization.

Proof of Lemma 3.A.10:

Consider the magnitude of the derivative $\frac{dg_k}{dt}$,

$$\left| \frac{dg_k}{dt} \right| = \left| \frac{1}{\sqrt{m}} \sum_{j=1}^n (f_j - y_j) \frac{c_k}{\|\mathbf{v}_k\|_2} \sigma(\mathbf{v}_k^\top \mathbf{x}_j) \right|.$$

Note

$$\left| \frac{c_k}{\|\mathbf{v}_k\|_2} \sigma(\mathbf{v}_k^\top \mathbf{x}_j) \right| = \left| \sigma\left(\frac{\mathbf{v}_k^\top \mathbf{x}_j}{\|\mathbf{v}_k\|_2} \right) \right| \leq 1$$

Thus applying Cauchy Schwartz

$$\left| \frac{dg_k(t)}{dt} \right| \leq \frac{2\sqrt{n}}{\sqrt{m}} \|\mathbf{f}(t) - \mathbf{y}\|_2 \leq \frac{2\sqrt{n}}{\sqrt{m}} \exp(-\omega t/2) \|\mathbf{f}(0) - \mathbf{y}\|_2,$$

and integrating from 0 to t yields

$$|g_k(t) - g_k(0)| \leq \int_0^t \left| \frac{dg_k(s)}{ds} \right| ds \leq \int_0^t \frac{2\sqrt{n}}{\sqrt{m}} \exp(-\omega s/2) \|\mathbf{f}(0) - \mathbf{y}\|_2 ds \leq \frac{4\sqrt{n} \|\mathbf{y} - \mathbf{f}(0)\|_2}{\sqrt{m}\omega}.$$

\square

Proof of Lemma 3.A.11:

Consider the i th entry of the network at initialization,

$$f_i(0) = \frac{1}{\sqrt{m}} \sum_{k=1}^m c_k \sigma \left(\frac{g_k \mathbf{v}_k^\top \mathbf{x}_i}{\|\mathbf{v}_k\|_2} \right).$$

Since the network is initialized randomly and m is taken to be large we apply concentration to bound $f_i(0)$ for each i . Define $z_k = c_k \sigma \left(\frac{g_k(0) \mathbf{v}_k(0)^\top \mathbf{x}_i}{\|\mathbf{v}_k(0)\|_2} \right)$. Note that z_k are independent sub-Gaussian random variables with

$$\|\mathbf{z}_k\|_\psi \leq \|N(0, 1)\|_\psi = C.$$

Here $\|\cdot\|_\psi$ denotes the 2-sub-Gaussian norm, (see [Ver18] for example). Applying Hoeffding's inequality bounds $f_i(0)$ as

$$\begin{aligned} \mathbb{P}(|\sqrt{m} f_i(0)| > t) &\leq 2 \exp \left(- \frac{t^2/2}{\sum_{k=1}^m \|\mathbf{z}_k\|_{\psi_2}} \right) \\ &= 2 \exp \left(\frac{-t^2}{2mC} \right). \end{aligned}$$

Which gives with probability $1 - \delta/n$ that

$$|f_i(0)| \leq \tilde{C} \sqrt{\log(n/\delta)}.$$

Now with probability $1 - \delta$ we have that, for each i ,

$$|f_i(0) - y_i| \leq |y_i| + \tilde{C} \sqrt{\log(n/\delta)} \leq C_2 \sqrt{\log(n/\delta)}.$$

Since $y_i = O(1)$. Hence, with probability $1 - \delta$,

$$\|\mathbf{f}(0) - \mathbf{y}\|_2 \leq C \sqrt{n \log(n/\delta)}.$$

□

Proof of Lemma 3.A.12:

At initialization $\mathbf{v}_k \sim N(0, \beta^2 \mathbf{I})$ so the norm behaves like $\|\mathbf{v}_k(0)\|_2^2 \sim \beta^2 \chi_d$. The cumulative density of a chi-squared distribution with d degrees of freedom behaves like $F(x) = \Theta(x^{d/2})$

for small x so we have that with probability $1 - \frac{\delta}{m}$, that $\|\mathbf{v}_k(0)\|_2 \geq \beta(m/\delta)^{\frac{1}{d}}$ where d is the input dimension. Applying a union bound, with probability $1 - \delta$, for all $1 \leq k \leq m$,

$$\frac{1}{\|\mathbf{v}_k(0)\|_2} \leq \frac{(m/\delta)^{1/d}}{\beta}.$$

Now by (3.5) for $t \geq 0$, $\|\mathbf{v}_k(t)\|_2 \geq \|\mathbf{v}_k(0)\|_2$ so

$$\frac{1}{\|\mathbf{v}_k(t)\|_2} \leq \frac{1}{\|\mathbf{v}_k(0)\|_2} \leq \frac{(m/\delta)^{1/d}}{\beta}.$$

□

3.A.3 Convergence proof for finite step-size training

The general technique of proof for gradient flow extends to finite-step gradient descent. Nonetheless, proving convergence for WeightNorm gradient descent exhibits additional complexities arising from the discrete updates and joint training with the new parameterization (3.2). We first introduce some needed notation.

Define $S_i(R)$ as the set of indices $k \in [m]$ corresponding to neurons that are close to the activity boundary of ReLU at initialization for a data point \mathbf{x}_i ,

$$S_i(R) := \{k \in [m] : \exists \mathbf{v} \text{ with } \|\mathbf{v} - \mathbf{v}_k(0)\|_2 \leq R \text{ and } \mathbf{1}_{ik}(0) \neq \mathbf{1}\{\mathbf{v}^\top \mathbf{x}_i \geq 0\}\}.$$

We upper bound the cardinality of $|S_i(R)|$ with high probability.

Lemma 3.A.14. *With probability $1 - \delta$, we have that for all i*

$$|S_i(R)| \leq \frac{\sqrt{2}mR}{\sqrt{\pi}\beta} + \frac{16 \log(n/\delta)}{3}.$$

Next we review some additional lemmas needed for the proof of Theorems 3.5.2, 3.5.3. Analogous to Lemmas 3.A.9, 3.A.10, we bound the finite-step parameter trajectories in Lemmas 3.A.15, 3.A.16.

Lemma 3.A.15. *Suppose the norm of $\|\mathbf{f}(s) - \mathbf{y}\|_2^2$ decreases linearly for some convergence rate ω during gradient descent training for all iteration steps $s = 0, 1, \dots, K$ with step-size η as $\|\mathbf{f}(s) - \mathbf{y}\|_2^2 \leq (1 - \frac{\eta\omega}{2})^s \|\mathbf{f}(0) - \mathbf{y}\|_2^2$. Then for each k we have*

$$|g_k(s) - g_k(0)| \leq \frac{4\sqrt{n}\|\mathbf{f}(0) - \mathbf{y}\|_2}{\sqrt{m}\omega}$$

for iterations $s = 0, 1, \dots, K + 1$.

Lemma 3.A.16. *Under the assumptions of Lemma 3.A.15, suppose in addition that $|g_k(s) - g_k(0)| \leq 1/(m/\delta)^{1/d}$ for all iterations steps $s = 0, 1, \dots, K$. Then for each k ,*

$$\|\mathbf{v}_k(s) - \mathbf{v}_k(0)\|_2 \leq \frac{8\sqrt{n}\|\mathbf{f}(0) - \mathbf{y}\|_2}{\beta\sqrt{m}\omega}$$

for $s = 0, 1, \dots, K + 1$.

To prove linear rate of convergence we analyze the $s + 1$ iterate error $\|\mathbf{f}(s + 1) - \mathbf{y}\|_2$ relative to that of the s iterate, $\|\mathbf{f}(s) - \mathbf{y}\|_2$. Consider the network's coordinate-wise difference in output between iterations, $f_i(s + 1) - f_i(s)$, writing this explicitly based on gradient descent updates yields

$$f_i(s + 1) - f_i(s) = \frac{1}{\sqrt{m}} \sum_{k=1}^m \frac{c_k g_k(s + 1)}{\|\mathbf{v}_k(s + 1)\|_2} \sigma(\mathbf{v}_k(s + 1)^\top \mathbf{x}_i) - \frac{c_k g_k(s)}{\|\mathbf{v}_k(s)\|_2} \sigma(\mathbf{v}_k(s)^\top \mathbf{x}_i). \quad (3.23)$$

We now decompose the summand in (3.23) looking at the updates in each layer, $f_i(s + 1) - f_i(s) = a_i(s) + b_i(s)$ with

$$a_i(s) = \frac{1}{\sqrt{m}} \sum_{k=1}^m \frac{c_k g_k(s + 1)}{\|\mathbf{v}_k(s + 1)\|_2} \sigma(\mathbf{v}_k(s)^\top \mathbf{x}_i) - \frac{c_k g_k(s)}{\|\mathbf{v}_k(s)\|_2} \sigma(\mathbf{v}_k(s)^\top \mathbf{x}_i),$$

$$b_i(s) = \frac{1}{\sqrt{m}} \sum_{k=1}^m \frac{c_k g_k(s + 1)}{\|\mathbf{v}_k(s + 1)\|_2} (\sigma(\mathbf{v}_k(s + 1)^\top \mathbf{x}_i) - \sigma(\mathbf{v}_k(s)^\top \mathbf{x}_i)).$$

Further, each layer summand is then subdivided into a primary term and a residual. $a_i(s)$, corresponding to the difference in the first layer $\left(\frac{c_k g_k(s + 1)}{\|\mathbf{v}_k(s + 1)\|_2} - \frac{c_k g_k(s)}{\|\mathbf{v}_k(s)\|_2} \right)$, is subdivided into $a_i^I(s)$ and $a_i^{II}(s)$ as follows:

$$a_i^I(s) = \frac{1}{\sqrt{m}} \sum_{k=1}^m \left(\frac{c_k g_k(s + 1)}{\|\mathbf{v}_k(s)\|_2} - \frac{c_k g_k(s)}{\|\mathbf{v}_k(s)\|_2} \right) \sigma(\mathbf{v}_k(s)^\top \mathbf{x}_i), \quad (3.24)$$

$$a_i^{II}(s) = \frac{1}{\sqrt{m}} \sum_{k=1}^m \left(\frac{c_k g_k(s + 1)}{\|\mathbf{v}_k(s + 1)\|_2} - \frac{c_k g_k(s + 1)}{\|\mathbf{v}_k(s)\|_2} \right) \sigma(\mathbf{v}_k(s)^\top \mathbf{x}_i). \quad (3.25)$$

$b_i(s)$ is sub-divided based on the indices in the set S_i that monitor the changes of the rectifiers. For now, $S_i = S_i(R)$ with R to be set later in the proof. $b_i(s)$ is partitioned to summands in the set S_i and the complement set,

$$\begin{aligned} b_i^I(s) &= \frac{1}{\sqrt{m}} \sum_{k \notin S_i} \frac{c_k g_k(s+1)}{\|\mathbf{v}_k(s+1)\|_2} (\sigma(\mathbf{v}_k(s+1)^\top \mathbf{x}_i) - \sigma(\mathbf{v}_k(s)^\top \mathbf{x}_i)), \\ b_i^{II}(s) &= \frac{1}{\sqrt{m}} \sum_{k \in S_i} \frac{c_k g_k(s+1)}{\|\mathbf{v}_k(s+1)\|_2} (\sigma(\mathbf{v}_k(s+1)^\top \mathbf{x}_i) - \sigma(\mathbf{v}_k(s)^\top \mathbf{x}_i)). \end{aligned}$$

With this sub-division in mind, the terms corresponding to convergence are $\mathbf{a}^I(s), \mathbf{b}^I(s)$ whereas $\mathbf{a}^{II}(s), \mathbf{b}^{II}(s)$ are residuals that are the result of discretization. We define the primary and residual vectors $\mathbf{p}(s), \mathbf{r}(s)$ as

$$\mathbf{p}(s) = \frac{\mathbf{a}^I(s) + \mathbf{b}^I(s)}{\eta}, \quad \mathbf{r}(s) = \frac{\mathbf{a}^{II}(s) + \mathbf{b}^{II}(s)}{\eta}. \quad (3.26)$$

If the residual $\mathbf{r}(s)$ is sufficiently small and $\mathbf{p}(s)$ may be written as $\mathbf{p}(s) = -\mathbf{\Lambda}(s)(\mathbf{f}(s) - \mathbf{y})$ for some iteration dependent evolution matrix $\mathbf{\Lambda}(s)$ that has

$$\lambda_{\min}(\mathbf{\Lambda}(s)) = \omega/2 \quad (3.27)$$

for $\omega > 0$ then the neural network (3.2) converges linearly when trained with WeightNorm gradient descent of step size η . We formalize the condition on $\mathbf{r}(s)$ below and later derive the conditions on the over-parametrization (m) ensuring that $\mathbf{r}(s)$ is sufficiently small.

Property 1. *Given a network from the class (3.2) initialized as in (3.6) and trained with gradient descent of step-size η , define the residual $\mathbf{r}(s)$ as in (3.26) and take ω as in (3.27). We specify the “residual condition” at iteration s as*

$$\|\mathbf{r}(s)\|_2 \leq c\omega \|\mathbf{f}(s) - \mathbf{y}\|_2$$

for a sufficiently small constant $c > 0$ independent of the data or initialization.

Here we present Theorem 3.A.17 which is the backbone of Theorems 3.5.2 and 3.5.3.

Theorem 3.A.17. *Suppose a network from the class (3.2) is trained via WeightNorm gradient descent with an evolution matrix $\mathbf{\Lambda}(s)$ as in (3.27) satisfying $\lambda_{\min}(\mathbf{\Lambda}(s)) \geq \omega/2$ for*

$s = 0, 1, \dots, K$. In addition if the data meets assumptions 1, 2, the step-size η of gradient descent satisfies $\eta \leq \frac{1}{3\|\mathbf{\Lambda}(s)\|_2}$ and that the residual $\mathbf{r}(s)$ defined in (3.26) satisfies Property 1 for $s = 0, 1, \dots, K$ then we have that

$$\|\mathbf{f}(s) - \mathbf{y}\|_2^2 \leq \left(1 - \frac{\eta\omega}{2}\right)^s \|\mathbf{f}(0) - \mathbf{y}\|_2^2$$

for $s = 0, 1, \dots, K$.

Proof of Theorem 3.A.17:

This proof provides the foundation for the main theorems. In the proof we also derive key bounds to be used in Theorems 3.5.2, 3.5.3. We use the decomposition we described above and consider again the difference between consecutive terms $\mathbf{f}(s+1) - \mathbf{f}(s)$,

$$f_i(s+1) - f_i(s) = \frac{1}{\sqrt{m}} \sum_{k=1}^m \frac{c_k g_k(s+1)}{\|\mathbf{v}_k(s+1)\|_2} \sigma(\mathbf{v}_k(s+1)^\top \mathbf{x}_i) - \frac{c_k g_k(s)}{\|\mathbf{v}_k(s)\|_2} \sigma(\mathbf{v}_k(s)^\top \mathbf{x}_i). \quad (3.28)$$

Following the decomposition introduced in (3.24), $a_i^I(s)$ is re-written in terms of $\mathbf{G}(s)$,

$$\begin{aligned} a_i^I(s) &= \frac{1}{\sqrt{m}} \sum_{k=1}^m \frac{c_k}{\|\mathbf{v}_k(s)\|_2} \left(-\eta \frac{\partial L(s)}{\partial g_k} \right) \sigma(\mathbf{v}_k(s)^\top \mathbf{x}_i) \\ &= -\frac{\eta}{m} \sum_{k=1}^m \frac{c_k}{\|\mathbf{v}_k(s)\|_2} \sum_{j=1}^n (f_j(s) - y_j) \frac{c_k}{\|\mathbf{v}_k(s)\|_2} \sigma(\mathbf{v}_k^\top(s) \mathbf{x}_j) \sigma(\mathbf{v}_k^\top(s) \mathbf{x}_i) \\ &= -\eta \sum_{j=1}^n (f_j(s) - y_j) \frac{1}{m} \sum_{k=1}^m (c_k)^2 \sigma\left(\frac{\mathbf{v}_k(s)^\top \mathbf{x}_i}{\|\mathbf{v}_k(s)\|_2}\right) \sigma\left(\frac{\mathbf{v}_k(s)^\top \mathbf{x}_j}{\|\mathbf{v}_k(s)\|_2}\right) \\ &= -\eta \sum_{j=1}^n (f_j(s) - y_j) \mathbf{G}_{ij}(s), \end{aligned}$$

where the first equality holds by the gradient update rule $g_k(s+1) = g_k(s) - \eta \nabla_{g_k} L(s)$. In this proof we also derive bounds on the residual terms of the decomposition which we will aid us in the proofs of Theorems 3.5.2, 3.5.3. $a_i^I(s)$ is the primary term of $a_i(s)$, now we bound the residual term $a_i^{II}(s)$. Recall $a_i^{II}(s)$ is written as

$$a_i^{II}(s) = \frac{1}{\sqrt{m}} \sum_{k=1}^m \left(\frac{c_k g_k(s+1)}{\|\mathbf{v}_k(s+1)\|_2} - \frac{c_k g_k(s+1)}{\|\mathbf{v}_k(s)\|_2} \right) \sigma(\mathbf{v}_k(s)^\top \mathbf{x}_i),$$

which corresponds to the difference in the normalization in the second layer. Since $\nabla_{\mathbf{v}_k} L(s)$

is orthogonal to $\mathbf{v}_k(s)$ we have that

$$\begin{aligned}
& c_k g_k(s+1) \left(\frac{1}{\|\mathbf{v}_k(s+1)\|_2} - \frac{1}{\|\mathbf{v}_k(s)\|_2} \right) \sigma(\mathbf{v}_k(s)^\top \mathbf{x}_i) \\
&= c_k g_k(s+1) \left(\frac{1}{\sqrt{\|\mathbf{v}_k(s)\|_2^2 + \eta^2 \|\nabla_{\mathbf{v}_k} L(s)\|_2^2}} - \frac{1}{\|\mathbf{v}_k(s)\|_2} \right) \sigma(\mathbf{v}_k(s)^\top \mathbf{x}_i) \\
&= \frac{-c_k g_k(s+1) \eta^2 \|\nabla_{\mathbf{v}_k} L(s)\|_2^2}{\|\mathbf{v}_k(s+1)\|_2 \|\mathbf{v}_k(s)\|_2 (\|\mathbf{v}_k(s)\|_2 + \|\mathbf{v}_k(s+1)\|_2)} \sigma(\mathbf{v}_k(s)^\top \mathbf{x}_i) \\
&\leq \frac{-c_k g_k(s+1) \eta^2 \|\nabla_{\mathbf{v}_k} L(s)\|_2^2}{2 \|\mathbf{v}_k(s)\|_2 \|\mathbf{v}_k(s+1)\|_2} \sigma \left(\frac{\mathbf{v}_k(s)^\top \mathbf{x}_i}{\|\mathbf{v}_k(s)\|_2} \right),
\end{aligned}$$

where the first equality above is by completing the square, and the inequality is due to the increasing magnitudes of $\|\mathbf{v}_k(s)\|_2$.

Since $0 \leq \sigma \left(\frac{\mathbf{v}_k(s)^\top \mathbf{x}_i}{\|\mathbf{v}_k(s)\|_2} \right) \leq 1$, the above can be bounded as

$$|a_i^{II}(s)| \leq \frac{1}{\sqrt{m}} \sum_{k=1}^m \left| \frac{g_k(s+1) \eta^2 \|\nabla_{\mathbf{v}_k} L(s)\|_2^2}{2 \|\mathbf{v}_k(s)\|_2 \|\mathbf{v}_k(s+1)\|_2} \right| \quad (3.29)$$

$$\leq \frac{1}{\sqrt{m}} \sum_{k=1}^m \frac{\eta^2 (1 + R_g(m/\delta)^{1/d})^3 n \|\mathbf{f}(s) - \mathbf{y}\|_2^2 (m/\delta)^{1/d}}{\beta^4 m} \quad (3.30)$$

$$= \frac{\eta^2 n (1 + R_g(m/\delta)^{1/d})^3 \|\mathbf{f}(s) - \mathbf{y}\|_2^2 (m/\delta)^{1/d}}{\beta^4 \sqrt{m}}. \quad (3.31)$$

The second inequality is the result of applying the bound in equation (3.41) on the gradient norm $\|\nabla_{\mathbf{v}_k} L(s)\|_2$ and using Lemma 3.A.12.

Next we analyze $b_i(s)$ and sub-divide it based on the sign changes of the rectifiers. Define the set $S_i := S_i(R)$ as in Lemma 3.A.14 with R taken to be such that $\|\mathbf{v}_k(s+1) - \mathbf{v}_k(0)\|_2 \leq R$ for all k . Take $b_i^{II}(s)$ as the sub-sum of $b_i(s)$ with indices k from the set S_i .

$b_i^I(s)$ corresponds to the sub-sum with the remaining indices. By the definition of S_i , for $k \notin S_i$ we have that $\mathbf{1}_{ik}(s+1) = \mathbf{1}_{ik}(s)$. This enables us to factor $\mathbf{1}_{ik}(s)$ and represent $b_i^I(s)$

as a Gram matrix similar to $\mathbf{V}(s)$ with a correction term from the missing indices in S_i .

$$\begin{aligned} b_i^I(s) &= -\frac{1}{\sqrt{m}} \sum_{k \notin S_i} \left(\frac{c_k g_k(s+1)}{\|\mathbf{v}_k(s+1)\|_2} \right) (\eta \langle \nabla_{\mathbf{v}_k} L(s), \mathbf{x}_i \rangle) \mathbb{1}_{ik}(s) \\ &= -\frac{\eta}{m} \sum_{k \notin S_i} \left(\frac{c_k g_k(s+1)}{\|\mathbf{v}_k(s+1)\|_2} \right) \left(\frac{c_k g_k(s)}{\|\mathbf{v}_k(s)\|_2} \right) \sum_{j=1}^n (f_j(s) - y_j) \mathbb{1}_{ik}(s) \mathbb{1}_{jk}(s) \langle \mathbf{x}_j^{\mathbf{v}_k(s)^\perp}, \mathbf{x}_i \rangle. \end{aligned}$$

Note that $\langle \mathbf{x}_j^{\mathbf{v}_k(s)^\perp}, \mathbf{x}_i \rangle = \langle \mathbf{x}_j^{\mathbf{v}_k(s)^\perp}, \mathbf{x}_i^{\mathbf{v}_k(s)^\perp} \rangle$ therefore,

$$b_i^I(s) = -\frac{\eta}{m} \sum_{k \notin S_i} \left(\frac{c_k g_k(s+1)}{\|\mathbf{v}_k(s+1)\|_2} \right) \left(\frac{c_k g_k(s)}{\|\mathbf{v}_k(s)\|_2} \right) \sum_{j=1}^n (f_j(s) - y_j) \mathbb{1}_{ik}(s) \mathbb{1}_{jk}(s) \langle \mathbf{x}_j^{\mathbf{v}_k(s)^\perp}, \mathbf{x}_i^{\mathbf{v}_k(s)^\perp} \rangle.$$

Define $\tilde{\mathbf{V}}(s)$ as

$$\tilde{\mathbf{V}}_{ij}(s) = \frac{1}{m} \sum_{k=1}^m \left(\frac{\beta c_k g_k(s+1)}{\|\mathbf{v}_k(s+1)\|_2} \right) \left(\frac{\beta c_k g_k(s)}{\|\mathbf{v}_k(s)\|_2} \right) \mathbb{1}_{jk}(s) \mathbb{1}_{ik}(s) \langle \mathbf{x}_i^{\mathbf{v}_k(s)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(s)^\perp} \rangle.$$

This matrix is identical to $\mathbf{V}(s)$ except for a modified scaling term $\left(\frac{c_k^2 g_k(s+1) g_k(s)}{\|\mathbf{v}_k(s)\|_2 \|\mathbf{v}_k(s+1)\|_2} \right)$. We note however that

$$\begin{aligned} \min \left(\left(\frac{c_k g_k(s)}{\|\mathbf{v}_k(s)\|_2} \right)^2, \left(\frac{c_k g_k(s+1)}{\|\mathbf{v}_k(s+1)\|_2} \right)^2 \right) &\leq \left(\frac{c_k g_k(s)}{\|\mathbf{v}_k(s)\|_2} \right) \left(\frac{c_k g_k(s+1)}{\|\mathbf{v}_k(s+1)\|_2} \right) \\ &\leq \max \left(\left(\frac{c_k g_k(s)}{\|\mathbf{v}_k(s)\|_2} \right)^2, \left(\frac{c_k g_k(s+1)}{\|\mathbf{v}_k(s+1)\|_2} \right)^2 \right) \end{aligned}$$

because $g_k(s), c_k^2$ are positive. Hence the matrix $\tilde{\mathbf{V}}(s)$ satisfies the hypothesis of Lemma 3.A.6 entirely. We write $b_i^I(s)$ as

$$b_i^I(s) = -\eta/\beta^2 \sum_{j=1}^n (f_j(s) - y_j) (\tilde{\mathbf{V}}_{ij}(s) - \tilde{\mathbf{V}}_{ij}^\perp(s)),$$

where we have defined

$$\tilde{\mathbf{V}}_{ij}^\perp(s) = \frac{1}{m} \sum_{k \in S_i} \left(\frac{\beta c_k g_k(s)}{\|\mathbf{v}_k(s)\|_2} \right) \left(\frac{\beta c_k g_k(s+1)}{\|\mathbf{v}_k(s+1)\|_2} \right) \mathbb{1}_{ik}(s) \mathbb{1}_{jk}(s) \langle \mathbf{x}_i^{\mathbf{v}_k(s)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(s)^\perp} \rangle. \quad (3.32)$$

We then bound the magnitude of each entry $\tilde{\mathbf{V}}_{ij}^\perp(s)$:

$$\tilde{\mathbf{V}}_{ij}^\perp(s) = \frac{1}{m} \sum_{k \in S_i} \left(\frac{\beta c_k g_k(s)}{\|\mathbf{v}_k(s)\|_2} \right) \left(\frac{\beta c_k g_k(s+1)}{\|\mathbf{v}_k(s+1)\|_2} \right) \mathbb{1}_{ik}(s) \mathbb{1}_{jk}(s) \langle \mathbf{x}_i^{\mathbf{v}_k(s)^\perp}, \mathbf{x}_j^{\mathbf{v}_k(s)^\perp} \rangle \quad (3.33)$$

$$\leq \frac{(1 + R_g(m/\delta)^{1/d})^2 |S_i|}{m}. \quad (3.34)$$

Lastly we bound the size of the residual term $b_i^{II}(s)$,

$$\begin{aligned} |b_i^{II}(s)| &= \left| -\frac{1}{\sqrt{m}} \sum_{k \in S_i} \frac{c_k g_k(s+1)}{\|\mathbf{v}_k(s+1)\|_2} \left(\sigma(\mathbf{v}_k(s+1)^\top \mathbf{x}_i) - \sigma(\mathbf{v}_k(s)^\top \mathbf{x}_i) \right) \right| \\ &\leq \frac{g_k(s+1) \eta |S_i| \cdot \|\nabla_{\mathbf{v}_k} L(s)\|_2}{\sqrt{m} \|\mathbf{v}_k(s+1)\|_2} \\ &\leq \frac{\eta |S_i| (1 + (m/\delta)^{1/d} R_g) \|\nabla_{\mathbf{v}_k} L(s)\|_2}{\beta \sqrt{m}}. \end{aligned}$$

Where we used the Lipschitz continuity of σ in the first bound, and took $R_g > 0$ that satisfies $|g_k(s+1) - g_k(0)| \leq R_g$ in the second inequality. Applying the bound (3.41),

$$|b_i^{II}(s)| \leq \frac{\eta |S_i| \sqrt{n} (1 + R_g (m/\delta)^{1/d})^2 \|\mathbf{f}(s) - \mathbf{y}\|_2}{\beta^2 m}. \quad (3.35)$$

The sum $\mathbf{f}(s+1) - \mathbf{f}(s) = \mathbf{a}^I(s) + \mathbf{a}^{II}(s) + \mathbf{b}^I(s) + \mathbf{b}^{II}(s)$ is separated into the primary term $\eta \mathbf{p}(s) = \mathbf{a}_I(s) + \mathbf{b}_I(s)$ and the residual term $\eta \mathbf{r}(s) = \mathbf{a}_{II}(s) + \mathbf{b}_{II}(s)$ which is a result of the discretization. With this, the evolution matrix $\mathbf{\Lambda}(s)$ in (3.27) is re-defined as

$$\mathbf{\Lambda}(s) := \mathbf{G}(s) + \frac{\tilde{\mathbf{V}}(s) - \tilde{\mathbf{V}}^1(s)}{\beta^2}$$

and

$$\mathbf{f}(s+1) - \mathbf{f}(s) = -\eta \mathbf{\Lambda}(s) (\mathbf{f}(s) - \mathbf{y}) + \eta \mathbf{r}(s).$$

Now we compare $\|\mathbf{f}(s+1) - \mathbf{y}\|_2^2$ with $\|\mathbf{f}(s) - \mathbf{y}\|_2^2$,

$$\begin{aligned} \|\mathbf{f}(s+1) - \mathbf{y}\|_2^2 &= \|\mathbf{f}(s+1) - \mathbf{f}(s) + \mathbf{f}(s) - \mathbf{y}\|_2^2 \\ &= \|\mathbf{f}(s) - \mathbf{y}\|_2^2 + 2 \langle \mathbf{f}(s+1) - \mathbf{f}(s), \mathbf{f}(s) - \mathbf{y} \rangle \\ &\quad + \langle \mathbf{f}(s+1) - \mathbf{f}(s), \mathbf{f}(s+1) - \mathbf{f}(s) \rangle. \end{aligned}$$

Substituting

$$\mathbf{f}(s+1) - \mathbf{f}(s) = \mathbf{a}^I(s) + \mathbf{b}^I(s) + \mathbf{a}^{II}(s) + \mathbf{b}^{II}(s) = -\eta \mathbf{\Lambda}(s) (\mathbf{f}(s) - \mathbf{y}) + \eta \mathbf{r}(s)$$

we obtain

$$\begin{aligned}
\|\mathbf{f}(s+1) - \mathbf{y}\|_2^2 &= \|\mathbf{f}(s) - \mathbf{y}\|_2^2 + 2(-\eta\mathbf{\Lambda}(s)(\mathbf{f}(s) - \mathbf{y}) + \eta\mathbf{r}(s))^\top(\mathbf{f}(s) - \mathbf{y}) \\
&\quad + \eta^2(\mathbf{\Lambda}(s)(\mathbf{f}(s) - \mathbf{y}) - \mathbf{r}(s))^\top(\mathbf{\Lambda}(s)(\mathbf{f}(s) - \mathbf{y}) - \mathbf{r}(s)) \\
&\leq \|\mathbf{f}(s) - \mathbf{y}\|_2^2 + (\mathbf{f}(s) - \mathbf{y})^\top(-\eta\mathbf{\Lambda}(s) + \eta^2\mathbf{\Lambda}^2(s))(\mathbf{f}(s) - \mathbf{y}) \\
&\quad + \eta\mathbf{r}(s)^\top(\mathbf{I} - \eta\mathbf{\Lambda}(s))(\mathbf{f}(s) - \mathbf{y}) + \eta^2\|\mathbf{r}(s)\|_2^2.
\end{aligned}$$

Now as $\lambda_{\min}(\mathbf{\Lambda}(s)) \geq \omega/2$ and $\eta = \frac{1}{3\|\mathbf{\Lambda}(s)\|_2}$, we have that

$$\begin{aligned}
(\mathbf{f}(s) - \mathbf{y})^\top(-\eta\mathbf{\Lambda}(s) + \eta^2\mathbf{\Lambda}^2(s))(\mathbf{f}(s) - \mathbf{y}) &= -\eta(\mathbf{f}(s) - \mathbf{y})^\top(\mathbf{I} - \eta\mathbf{\Lambda}(s))\mathbf{\Lambda}(s)(\mathbf{f}(s) - \mathbf{y}) \\
&\leq -\frac{3\eta\omega}{8}\|\mathbf{f}(s) - \mathbf{y}\|_2^2.
\end{aligned}$$

Next we analyze the rest of the terms and group them as $\mathbf{q}(s)$,

$$\begin{aligned}
\mathbf{q}(s) &:= \eta\mathbf{r}(s)^\top(\mathbf{I} - \eta\mathbf{\Lambda}(s))(\mathbf{f}(s) - \mathbf{y}) + \eta^2\|\mathbf{r}(s)\|_2^2 \\
&\leq \eta\|\mathbf{r}(s)\|_2\|\mathbf{f}(s) - \mathbf{y}\|_2 + \eta^2\|\mathbf{r}(s)\|_2^2.
\end{aligned}$$

By Property 1 we have

$$\mathbf{q}(s) \leq \eta c\omega\|\mathbf{f}(s) - \mathbf{y}\|_2^2(1 + \eta c\omega) \leq 2c\eta\omega\|\mathbf{f}(s) - \mathbf{y}\|_2^2,$$

so that

$$\mathbf{q}(s) \leq c'\eta\omega\|\mathbf{f}(s) - \mathbf{y}\|_2^2,$$

for c' sufficiently small. Substituting, the difference $\mathbf{f}(s+1) - \mathbf{y}$ is bounded as

$$\begin{aligned}
\|\mathbf{f}(s+1) - \mathbf{y}\|_2^2 &\leq \|\mathbf{f}(s) - \mathbf{y}\|_2^2 - \eta\omega(1 - \eta\|\mathbf{\Lambda}(s)\|_2)\|\mathbf{f}(s) - \mathbf{y}\|_2^2 + c'\eta\omega\|\mathbf{f}(s) - \mathbf{y}\|_2^2 \\
&\leq (1 - \eta\omega(1 - \eta\|\mathbf{\Lambda}(s)\|_2) + c'\eta\omega)\|\mathbf{f}(s) - \mathbf{y}\|_2^2 \\
&\leq (1 - \eta\omega/2)\|\mathbf{f}(s) - \mathbf{y}\|_2^2,
\end{aligned}$$

for well chosen absolute constant c . Hence for each $s = 0, 1, \dots, K$,

$$\|\mathbf{f}(s+1) - \mathbf{y}\|_2^2 \leq (1 - \eta\omega/2)\|\mathbf{f}(s) - \mathbf{y}\|_2^2,$$

so the prediction error converges linearly. \square

In what comes next we prove the necessary conditions for Property 1, and define the appropriate ω for the \mathbf{V} and \mathbf{G} dominated regimes, in order to show $\lambda_{\min}(\mathbf{\Lambda}(s)) \geq \omega/2$.

Proof of Theorem 3.5.2:

To prove convergence we would like to apply Theorem 3.A.17 with $\omega/2 = \frac{\lambda_0}{2\beta^2}$. To do so we need to show that $m = \Omega(n^4 \log(n/\delta)/\lambda_0^4)$ guarantees that Property 1 holds and that $\lambda_{\min}(\mathbf{\Lambda}(s)) \geq \lambda_0/2\beta^2$. For finite-step gradient training, take

$$R_v = \frac{\beta\lambda_0}{192n(m/\delta)^{1/d}}, \quad R_g = \frac{\lambda_0}{96n(m/\delta)^{1/d}}. \quad (3.36)$$

Note the residual $\mathbf{r}(s)$ and the other terms $\mathbf{b}_I(s)$, $\mathbf{b}_{II}(s)$ depend on the sets S_i that we define here using R_v . We make the assumption that $\|\mathbf{v}_k(s) - \mathbf{v}_k(0)\|_2 \leq R_v$ and $|g_k(s) - g_k(0)| \leq R_g$ for all k and that $s = 0, 1, \dots, K+1$, this guarantees that $\mathbf{b}_I(s)$ and $\mathbf{\Lambda}(s)$ are well defined. Applying Lemmas 3.A.3, 3.A.6 with R_v, R_g defined above, we have that $\lambda_{\min}(\tilde{\mathbf{V}}(s)) \geq \frac{5\lambda_0}{8}$. Then the least eigenvalue of the evolution matrix $\mathbf{\Lambda}(s)$ is bounded below

$$\begin{aligned} \lambda_{\min}(\mathbf{\Lambda}(s)) &= \lambda_{\min}\left(\mathbf{G}(s) + \frac{\tilde{\mathbf{V}}(s) - \tilde{\mathbf{V}}^\perp(s)}{\beta^2}\right) \\ &\geq \lambda_{\min}\left(\frac{\tilde{\mathbf{V}}(s) - \tilde{\mathbf{V}}^\perp(s)}{\beta^2}\right) \\ &= \frac{\lambda_{\min}(\tilde{\mathbf{V}}(s) - \tilde{\mathbf{V}}^\perp(s))}{\beta^2} \\ &\geq \frac{5\lambda_0}{8\beta^2} - \frac{\|\tilde{\mathbf{V}}^\perp(s)\|_2}{\beta^2}. \end{aligned}$$

The first inequality holds since $\mathbf{G}(s) > 0$ and the last inequality is since $\lambda_{\min}(\tilde{\mathbf{V}}(s)) \geq \frac{5\lambda_0}{8}$.

To show $\lambda_{\min}(\mathbf{\Lambda}(s)) \geq \frac{\lambda_0}{2\beta^2}$ we bound $\|\tilde{\mathbf{V}}^\perp(s)\|_2 \leq \frac{\lambda_0}{8}$. By (3.33), we have

$$|\tilde{\mathbf{V}}_{ij}^\perp(s)| \leq \frac{(1 + R_g(m/\delta)^{1/d})|S_i|}{m} \leq (1 + R_g(m/\delta)^{1/d}) \left(\frac{\sqrt{2}\tilde{R}_v}{\sqrt{\pi}\beta} + \frac{16 \log(n/\delta)}{3m} \right).$$

Substituting R_v, R_g and m , a direct calculation shows that

$$|\tilde{\mathbf{V}}_{ij}^\perp(s)| \leq \frac{\lambda_0}{8n},$$

which yields

$$\|\tilde{\mathbf{V}}^\perp(s)\|_2 \leq \|\tilde{\mathbf{V}}^\perp(s)\|_F \leq \frac{\lambda_0}{8}.$$

Hence $\lambda_{\min}(\mathbf{\Lambda}(s)) \geq \frac{\lambda_0}{2\beta^2}$ for iterations $s = 0, 1, \dots, K$.

We proceed by showing the residual $\mathbf{r}(s)$ satisfies property 1. Recall $\mathbf{r}(s)$ is written as

$$\mathbf{r}(s) = \frac{\mathbf{a}^{II}(s)}{\eta} + \frac{\mathbf{b}^{II}(s)}{\eta}.$$

and Property 1 states that $\|\mathbf{r}(s)\|_2 \leq \frac{c\eta\lambda_0}{\beta^2}\|\mathbf{f}(s) - \mathbf{y}\|_2$ for sufficiently small absolute constant $c < 1$. This is equivalent to showing that both $\mathbf{a}^{II}(s)$, $\mathbf{b}^{II}(s)$ satisfy

$$\|\mathbf{a}^{II}(s)\|_2 \leq \frac{c\eta\lambda_0}{\beta^2}\|\mathbf{f}(s) - \mathbf{y}\|_2, \quad (3.37)$$

$$\|\mathbf{b}^{II}(s)\|_2 \leq \frac{c\eta\lambda_0}{\beta^2}\|\mathbf{f}(s) - \mathbf{y}\|_2. \quad (3.38)$$

We consider each term at turn. By (3.35),

$$\begin{aligned} \|\mathbf{b}^{II}(s)\|_2 &\leq \sqrt{n} \max_i b_i^{II}(s) \\ &\leq \max_i \frac{\eta n (1 + R_g(m/\delta)^{1/d})^2 |S_i| \|\mathbf{f}(s) - \mathbf{y}\|_2}{\beta^2 m} \\ &\leq \frac{C m R_v \eta n \|\mathbf{f}(s) - \mathbf{y}\|_2}{\beta^2 m} \\ &\leq \frac{\lambda_0 \eta \|\mathbf{f}(s) - \mathbf{y}\|_2}{\beta^2} \cdot n C R_v. \end{aligned}$$

In the above we used the values of R_v, R_g defined in (3.36) and applied Lemma 3.A.14 in the third inequality. Taking $m = \Omega(n^4 \log(n/\delta)/\lambda_0^4)$ with large enough constant yields

$$\|\mathbf{b}^{II}(s)\|_2 \leq \frac{c\lambda_0\eta\|\mathbf{f}(s) - \mathbf{y}\|_2}{\beta^2}.$$

Next we analogously bound $\|\mathbf{a}^{II}(s)\|$ via the bound (3.29),

$$\begin{aligned}
\|\mathbf{a}^{II}(s)\|_2 &\leq \sqrt{n} \max_i a_i^{II}(s) \\
&\leq \frac{\eta^2 n^{3/2} (1 + R_g(m/\delta)^{1/d})^3 \|\mathbf{f}(s) - \mathbf{y}\|_2^2 (m/\delta)^{1/d}}{\beta^4 \sqrt{m}} \\
&\leq \frac{\eta \lambda_0 \|\mathbf{f}(s) - \mathbf{y}\|_2}{\beta^2} \cdot \frac{\eta (1 + R_g(m/\delta)^{1/d})^3 n^{3/2} \|\mathbf{f}(s) - \mathbf{y}\|_2 (m/\delta)^{1/d}}{\lambda_0 \beta^2 \sqrt{m}} \\
&\leq \frac{\eta \lambda_0 \|\mathbf{f}(s) - \mathbf{y}\|_2}{\beta^2} \cdot \frac{\eta}{\beta^2} \cdot \frac{C n^2 \sqrt{\log(n/\delta)}}{\lambda_0 \sqrt{m}} \\
&\leq c \eta \omega \|\mathbf{f}(s) - \mathbf{y}\|_2.
\end{aligned}$$

In the above we applied Lemma 3.A.11 once again. The last inequality holds since $m = \Omega(n^4 \log(n/\delta)/\lambda_0^4)$ and $\eta = O\left(\frac{\beta^2}{\|\mathbf{v}(s)\|_2}\right)$, hence $\mathbf{r}(s)$ satisfies Property 1. Now since Theorem 3.A.17 holds with $\omega = \lambda_0/\beta^2$ we have that the maximum parameter trajectories are bounded as $\|\mathbf{v}_k(s) - \mathbf{v}_k(0)\|_2 \leq R_v$ and $\|g_k(s) - g_k(0)\| \leq R_g$ for all k and every iteration $s = 0, 1, \dots, K + 1$ via Lemmas 3.A.15, 3.A.16.

To finish the proof, we apply the same contradiction argument as in Theorems 3.A.1, 3.A.2, taking the first iteration $s = K_0$ where one of Lemmas 3.A.15, 3.A.16 does not hold. We note that $K_0 > 0$ and by the definition of K_0 , for $s = 0, 1, \dots, K_0 - 1$ the Lemmas 3.A.15, 3.A.16 hold which implies that by the argument above we reach linear convergence in iteration $s = K_0$. This contradicts one of Lemmas 3.A.15, 3.A.16 which gives the desired contradiction, so we conclude that we have linear convergence with rate $\lambda_0/2\beta^2$ for all iterations. \square

Proof of Theorem 3.5.3:

For \mathbf{G} -dominated convergence, we follow the same steps as in the proof of Theorem 3.5.2. We redefine the trajectory constants for the finite step case

$$\tilde{R}_v := \frac{\sqrt{2\pi}\beta\mu_0}{64n(m/\delta)^{1/d}}, \quad R_g := \frac{\mu_0}{48n(m/\delta)^{1/d}}.$$

To use Theorem 3.A.17 we need to show that $m = \Omega(n^4 \log(n/\delta)/\beta^4 \mu_0^4)$ guarantees Property 1, and that $\lambda_{\min}(\mathbf{\Lambda}(s)) \geq \mu_0/2$. We again note that the residual $\mathbf{r}(s)$ and $\mathbf{b}_I(s), \mathbf{b}_{II}(s)$ depend on the sets S_i that we define here using \tilde{R}_v above as $S_i := S_i(\tilde{R}_v)$.

We start by showing the property on the least eigenvalue. We make the assumption that we have linear convergence with $\omega/2 = \mu_0/2$ and step-size η for iterations $s = 0, \dots, K$ so that Lemmas 3.A.15, 3.A.16 hold. Via an analogous analysis of the continuous case we reach that $m = \Omega(n^4 \log(n/\delta)/\mu_0^4 \beta^4)$ implies

$$\|\mathbf{v}_k(s) - \mathbf{v}_k(0)\|_2 \leq \frac{16\beta\sqrt{n}\|\mathbf{f}(0) - \mathbf{y}\|_2}{\beta\sqrt{m}\mu_0} \leq \tilde{R}_v, \quad |g_k(s) - g_k(0)| \leq \frac{8\sqrt{n}\|\mathbf{f}(0) - \mathbf{y}\|_2}{\sqrt{m}\mu_0} \leq R_g.$$

for $s = 0, \dots, K+1$ by Lemmas 3.A.15, 3.A.16 and that $\mathbf{\Lambda}(s), \mathbf{b}_I(s)$ are well defined. Using the bounds on the parameter trajectories, Lemma 3.A.7 and \tilde{R}_v defined above yield $\lambda_{\min}(\mathbf{G}(s)) \geq \frac{5\mu_0}{8}$. The least eigenvalue of the evolution matrix $\mathbf{\Lambda}(s)$ is bounded below as

$$\begin{aligned} \lambda_{\min}(\mathbf{\Lambda}(s)) &= \lambda_{\min}\left(\mathbf{G}(s) + \frac{\tilde{\mathbf{V}}(s) - \tilde{\mathbf{V}}^\perp(s)}{\beta^2}\right) \\ &\geq \lambda_{\min}(\mathbf{G}(s)) - \|\tilde{\mathbf{V}}^\perp(s)\|_2 \end{aligned}$$

since $\tilde{\mathbf{V}}(s) > 0$ and $\beta \geq 1$. We bound the spectral norm of $\|\tilde{\mathbf{V}}^\perp(s)\|_2$, for each entry i, j we have by (3.33) that

$$\begin{aligned} |\tilde{\mathbf{V}}_{ij}^\perp(s)| &\leq \frac{(1 + R_g(m/\delta)^{1/d})|S_i|}{m} \\ &\leq (1 + R_g(m/\delta)^{1/d})\left(\frac{\sqrt{2}\tilde{R}_v}{\sqrt{\pi}\beta} + \frac{16\log(n/\delta)}{3m}\right) \\ &\leq \frac{8\tilde{R}_v}{\sqrt{2\pi}\beta} \\ &\leq \frac{\mu_0}{8n}. \end{aligned}$$

where in the above inequalities we used our bounds on \tilde{R}_v, R_g and m . Then the spectral norm is bounded as

$$\|\tilde{\mathbf{V}}^\perp(s)\|_2 \leq \|\tilde{\mathbf{V}}^\perp(s)\|_F \leq \mu_0/8.$$

Hence we have that $\lambda_{\min}(\mathbf{\Lambda}(s)) \geq \mu_0/2$ for $s = 0, 1, \dots, K$.

Next we show the residual $\mathbf{r}(s)$ satisfies Property 1. Recall $\mathbf{r}(s)$ is written as

$$\mathbf{r}(s) = \frac{\mathbf{a}^{II}(s)}{\eta} + \frac{\mathbf{b}^{II}(s)}{\eta}.$$

Property 1 states the condition $\|\mathbf{r}(s)\|_2 \leq c\omega\eta\|\mathbf{f}(s) - \mathbf{y}\|_2$ for sufficiently small $c < 1$ with $\omega = \mu_0$. This is equivalent to showing that both $\mathbf{a}^{II}(s)$, $\mathbf{b}^{II}(s)$ satisfy that

$$\|\mathbf{a}^{II}(s)\|_2 \leq c\eta\mu_0\|\mathbf{f}(s) - \mathbf{y}\|_2, \quad (3.39)$$

$$\|\mathbf{b}^{II}(s)\|_2 \leq c\eta\mu_0\|\mathbf{f}(s) - \mathbf{y}\|_2, \quad (3.40)$$

for sufficiently small absolute constant c . For $\mathbf{b}_{II}(s)$ we have that (3.35) gives

$$\begin{aligned} \|\mathbf{b}^{II}(s)\|_2 &\leq \sqrt{n} \max_i b_i^{II}(s) \\ &\leq \max_i \frac{\eta(1 + R_g(m/\delta)^{1/d})^2 |S_i| n \|\mathbf{f}(s) - \mathbf{y}\|_2}{\beta^2 m}. \end{aligned}$$

Next applying Lemmas 3.A.14 and 3.A.11 in turn yields

$$\begin{aligned} &\leq \frac{Cm\tilde{R}_v\eta n \|\mathbf{f}(s) - \mathbf{y}\|_2}{\beta^2 m} \\ &\leq \eta\mu_0\|\mathbf{f}(s) - \mathbf{y}\|_2 \frac{\tilde{R}_v}{n\beta^2}. \end{aligned}$$

Substituting $m = \Omega(n^4 \log(n/\delta)/\mu_0^4 \beta^4)$ for a large enough constant and R_v we get

$$\|\mathbf{b}^{II}(s)\|_2 \leq c\eta\mu_0\|\mathbf{f}(s) - \mathbf{y}\|_2.$$

Analogously we bound $\|\mathbf{a}^{II}(s)\|_2$ using (3.29),

$$\begin{aligned} \|\mathbf{a}^{II}(s)\|_2 &\leq \sqrt{n} \max_i a_i^{II}(s) \\ &\leq \frac{\eta^2 n^{3/2} (1 + R_g(m/\delta)^{1/d})^3 \|\mathbf{f}(s) - \mathbf{y}\|_2^2 (m/\delta)^{1/d}}{\beta^4 \sqrt{m}} \\ &\leq \eta\mu_0\|\mathbf{f}(s) - \mathbf{y}\|_2 \cdot \frac{\eta(1 + R_g(m/\delta)^{1/d})^3 n^{3/2} \|\mathbf{f}(s) - \mathbf{y}\|_2 (m/\delta)^{1/d}}{\mu_0 \beta^4 \sqrt{m}} \\ &\leq \eta\mu_0\|\mathbf{f}(s) - \mathbf{y}\|_2 \cdot \frac{\eta}{\beta^2} \cdot \frac{Cn^2 \sqrt{\log(n/\delta)}}{\beta^2 \mu_0^2 \sqrt{m}} \\ &\leq c\eta\mu_0\|\mathbf{f}(s) - \mathbf{y}\|_2. \end{aligned}$$

Where we have used Lemma 3.A.11 in the third inequality and substituted $m = \Omega(n^4 \log(n/\delta)/\beta^4 \mu_0^4)$ noting that $\eta = O\left(\frac{1}{\|\mathbf{\Lambda}(s)\|_2}\right)$ and that $\beta \geq 1$ in the last inequality. Therefore we have that $\mathbf{r}(s)$ satisfies Property 1 so that Theorem 3.A.17 holds. By the same contradiction argument as in Theorem 3.5.2 we have that this holds for all iterations. \square

3.A.3.1 Proofs of supporting lemmas for finite step-size convergence

Proof of Proposition 8:

The proof of proposition 2, follows the proofs of Theorems 3.5.2, 3.5.3, and relies on Theorem 3.A.17. In particular for each $\beta > 0$ at initialization, take $\omega_\beta(s) = \lambda_{\min}(\mathbf{\Lambda}(s))$ and define the auxiliary $\omega_{\beta,0} = \lambda_{\min}(\mathbf{V}^\infty/\beta^2 + \mathbf{G}^\infty)$. Then we have that

$$\omega_{\beta,0} \geq \lambda_0/\beta^2 + \mu_0 > 0.$$

Hence, by the same arguments of Theorem 4.1, 4.2 for $\omega_\beta(s)$ if $m = (n^4 \log(n/\delta)/\beta^4 \omega_{\beta,0}^4)$, then we have that the conditions of Theorem 3.A.17 are satisfied, namely, $\lambda(s) \geq \frac{\lambda_0}{2}$ and $\mu(s) \geq \frac{\mu_0}{2}$. Taking $\eta_\beta = O\left(\frac{1}{\|\mathbf{\Lambda}(s)\|_2}\right)$, then the required step-size for convergence is satisfied. This follows from the same argument of Theorems 3.5.2, 3.5.3 and depends on the fact that $\|\mathbf{\Lambda}(s) - \mathbf{\Lambda}(0)\|_2 \leq \frac{1}{\beta^2} \|\mathbf{V}(s) - \mathbf{V}^\infty(0)\|_2 + \|\mathbf{G}(s) - \mathbf{G}(0)\|_2$. Now we consider the term, $\beta\omega_{\beta,0}$. For $\beta = 1$,

$$\beta\omega_{\beta,0} = \lambda_{\min}(\mathbf{H}^\infty).$$

Which matches the results of un-normalized convergence. In general, we have that

$$\beta\omega_{\beta,0} \geq \beta(\lambda_0/\beta^2 + \mu_0) \geq \min\{\lambda_0, \mu_0\}.$$

Therefore the bound on m is taken to be independent of β as $m = \Omega\left(\frac{n^4 \log(n/\delta)}{\min\{\mu_0^4, \lambda_0^4\}}\right)$ which simplifies the presentation. Now for each β the effective convergence rate is dictated by the least eigenvalue ω_β and the allowed step-size η_β as,

$$\left(1 - \eta_\beta \omega_\beta\right).$$

Then taking $\beta^* = \operatorname{argmin}_{\beta > 0} (1 - \eta_\beta \omega_\beta)$ we have that

$$(1 - \eta_{\beta^*} \omega_{\beta^*}) \leq (1 - \eta_1 \omega_1).$$

which corresponds to the un-normalized convergence rate. Therefore as compared with un-normalized training we have that for β^* , WN enables a faster convergence rate. \square

Proof of Lemma 3.A.14:

Fix R , without the loss of generality we write S_i for $S_i(R)$. For each k , $\mathbf{v}_k(0)$ is initialized independently via $\sim N(0, \beta^2 \mathbf{I})$, and for a given k , the event $\mathbb{1}_{i_k}(0) \neq \mathbb{1}\{\mathbf{v}^\top \mathbf{x}_i \geq 0\}$ corresponds to $|\mathbf{v}_k(0)^\top \mathbf{x}_i| \leq R$. Since $\|\mathbf{x}_i\|_2 = 1$, $\mathbf{v}_k(0)^\top \mathbf{x}_i \sim N(0, \beta^2)$. Denoting the event that an index $k \in S_i$ as $A_{i,k}$, we have

$$\mathbb{P}(A_{i,k}) \leq \frac{2R}{\beta\sqrt{2\pi}}.$$

Next the cardinality of S_i is written as

$$|S_i| = \sum_{k=1}^m \mathbb{1}_{A_{i,k}}.$$

Applying Lemma 3.A.13, with probability $1 - \delta/n$,

$$|S_i| \leq \frac{2mR}{\beta\sqrt{2\pi}} + \frac{16 \log(n/\delta)}{3}.$$

Taking a union bound, with probability $1 - \delta$, for all i we have that

$$|S_i| \leq \frac{2mR}{\beta\sqrt{2\pi}} + \frac{16 \log(n/\delta)}{3}.$$

□

Proof of Lemma 3.A.15:

To show this we bound the difference $g_k(s) - g_k(0)$ as the sum of the iteration updates. Each update is written as

$$\left| \frac{\partial L(s)}{\partial g_k} \right| = \left| \frac{1}{\sqrt{m}} \sum_{i=1}^n (f_i(s) - y_i) \frac{c_k}{\|\mathbf{v}_k(s)\|_2} \sigma(\mathbf{v}_k(s)^\top \mathbf{x}_i) \right|.$$

As $\left| c_k \sigma\left(\frac{\mathbf{v}_k(s)^\top \mathbf{x}_i}{\|\mathbf{v}_k(s)\|_2}\right) \right| \leq 1$,

$$\left| \frac{\partial L(s)}{\partial g_k} \right| \leq \frac{1}{\sqrt{m}} \sum_i^n |f_i(s) - y_i| \leq \frac{\sqrt{n}}{\sqrt{m}} \|\mathbf{f}(s) - \mathbf{y}\|_2.$$

By the assumption in the statement of the lemma,

$$\left| \frac{\partial L(s)}{\partial g_k} \right| \leq \frac{\sqrt{n} (1 - \frac{\eta\omega}{2})^{s/2} \|\mathbf{f}(0) - \mathbf{y}\|_2}{\sqrt{m}}.$$

Hence bounding the difference by the sum of the gradient updates:

$$|g_k(K+1) - g_k(0)| \leq \eta \sum_{s=0}^K \left| \frac{\partial L(s)}{\partial g_k} \right| \leq \frac{4\eta\sqrt{n}\|\mathbf{f}(0) - \mathbf{y}\|_2}{\sqrt{m}} \sum_{s=0}^K \left(1 - \frac{\eta\omega}{2}\right)^{s/2}.$$

The last term yields a geometric series that is bounded as

$$\frac{1}{1 - \sqrt{1 - \frac{\eta\omega}{2}}} \leq \frac{4}{\eta\omega},$$

Hence

$$|g_k(K+1) - g_k(0)| \leq \frac{4\sqrt{n}\|\mathbf{f}(0) - \mathbf{y}\|_2}{\omega\sqrt{m}}.$$

□

Proof of Lemma 3.A.16:

To show this we write $\mathbf{v}_k(s)$ as the sum of gradient updates and the initial weight $\mathbf{v}_k(0)$. Consider the norm of the gradient of the loss with respect to \mathbf{v}_k ,

$$\|\nabla_{\mathbf{v}_k} L(s)\|_2 = \left\| \frac{1}{\sqrt{m}} \sum_{i=1}^n (f_i(s) - y_i) \frac{c_k g_k(s)}{\|\mathbf{v}_k(s)\|_2} \mathbf{1}_{ik}(s) \mathbf{x}_i^{\mathbf{v}_k(s)\perp} \right\|_2.$$

Since $\|\mathbf{v}_k(s)\|_2 \geq \|\mathbf{v}_k(0)\|_2 \geq \beta(\delta/m)^{1/d}$ with probability $1 - \delta$ over the initialization, applying Cauchy Schwartz's inequality gives

$$\|\nabla_{\mathbf{v}_k} L(s)\|_2 \leq \frac{(1 + R_g(m/\delta)^{1/d})\sqrt{n}\|\mathbf{f}(s) - \mathbf{y}\|_2}{\beta\sqrt{m}}. \quad (3.41)$$

By the assumption on $\|\mathbf{f}(s) - \mathbf{y}\|_2$ this gives

$$\|\nabla_{\mathbf{v}_k} L(s)\|_2 \leq \frac{2\sqrt{n}(1 - \frac{\eta\omega}{2})^{s/2}\|\mathbf{f}(0) - \mathbf{y}\|_2}{\beta\sqrt{m}}.$$

Hence bounding the parameter trajectory by the sum of the gradient updates:

$$\|\mathbf{v}_k(K+1) - \mathbf{v}_k(0)\|_2 \leq \eta \sum_{s=0}^K \|\nabla_{\mathbf{v}_k} L(s)\|_2 \leq \frac{2\sqrt{n}\|\mathbf{f}(0) - \mathbf{y}\|_2}{\beta\sqrt{m}} \sum_{s=1}^K \left(1 - \frac{\eta\omega}{2}\right)^{s/2}$$

yields a geometric series. Now the series is bounded as

$$\frac{1}{1 - \sqrt{1 - \frac{\eta\omega}{2}}} \leq \frac{4}{\eta\omega},$$

which gives

$$\|\mathbf{v}_k(K+1) - \mathbf{v}_k(0)\|_2 \leq \frac{8\sqrt{n}\|\mathbf{f}(0) - \mathbf{y}\|_2}{\beta\sqrt{m\omega}}.$$

□

Final remarks for Part I

As the field of deep learning matures, data, algorithms, and hardware are continually advancing rapidly. Implementation frameworks such as PyTorch, TensorFlow are highly efficient and extensible, which allows for fast prototyping and experimentation of new architectures that are available daily on GitHub. From the perspective of data, datasets and benchmarks are ever-growing and are becoming more publicly available in a range of fields [BBN19]. In addition architectural and training discoveries such as the Transformer architecture [VSP17] and CLIP [RKH21] are redefining archetypal neural network. The availability and reproducibility of this science is leading to a spring of innovation and enables large strides on this subject field. Yet despite the incessant activity in the field, some things (currently) remain, in particular 1. the importance of catering and thoroughly understanding the data modalities used to learn patterns in modern models and 2. the optimization procedure used to train such models. Both pieces are currently indispensable in the ongoing revolution of deep learning.

In this part of the thesis, we focus on methods to improve the use of data in machine learning. In Chapter 1 we defined a notion of smoothness in models according to an alternative view of distance between data samples. The distance between data samples is the Wasserstein or Earth-Movers distance which represent each image datum as a mass distribution. In the chapter we illustrate the many benefits of using the notion of distance we present, named the Wasserstein Ground Metric. As we finish this part, we note that there are still a lot of questions on what are the correct way to represent visual data. Drawing inspiration from the visual system and the brain, the human visual system captures image representation using a rather sparse, foveated, and dynamically-stitched input representation, whereas in Deep Learning pixelated images serve as the common input. Further, deep learning models exhibit sensitivity to the input resolution and it has been shown that the resolution of images [TL19] has a large effect on state of the art results. While the input representation is not clear, we note that in many applications it is assumed implicitly or explicitly that the notion

of image similarity follows the distance defined by the Euclidean metric. We illustrate that this assumption is undesirable and illustrate the use of the Wasserstein Ground Metric as a metric that is 1) anisotropic in pixel space 2) depends on the location of measurement. We illustrate that the WGM enables semantic smoothness along variations such as translations and rotations. We note, that in the context of the Wasserstein metric on images, it still remains to identify how to define the ground topology in pixel space. Since with the WGM, different ground topologies defining distances between pixels would translate to different behaviors of the metric. In addition it would be interesting to identify a natural way of assigning such topologies especially under different image resolutions and modalities.

Our next area of focus was in probing and modifying the i.i.d. assumption imposed in most deep learning regimes through the foundational empirical risk minimization framework. In particular it is clear that for many applications the training set was collected with a certain bias which is not reflected in the test distribution or production settings. Given such disparity between the training set and desired distribution, we present a framework that selects optimal subsets of the training set and assigns distribution weights on the training set to end-to-end optimize a validation set. When a validation set is not available we utilized tools from cross-validation to derive tractable alternatives and illustrated the results by improving fine-grained classification tasks and more efficiently selecting samples to add to the training set. The formulation we present in Chapter 2 relies on model linearization for end-to-end differentiation, and the linearization that is applied in the experiments has the potential to be extended to full linearization of the model, this future direction still needs to be experimented. In general, Chapter 2 marks a transition from the empirical risk minimization approach of machine learning, to a more flexible framework where models are given more flexibility and feedback to select the data in their training set and optimize such training set selection end-to-end to learn more efficiently. The extension of learning from the empirical risk minimization to more flexible approaches, will enable models to learn more efficiently and more diverse tasks.

The “learning” in deep learning today amounts to parameter optimization, usually in the over-parameterized regime, and done with a first order algorithms such as SGD. In the last chapter of Part I, we analyzed such optimization under the different parameterization provided by normalization layers. Indeed, normalization layers in deep learning, are exactly parameterization that come to address the optimization problem. By applying a normalization step during the forward operation graph of the model, such layers enable improved speed of training and also better generalization. Because normalization layers, such as BatchNorm, WeightNorm, and LayerNorm facilitate learning via modified parameterization, they are widely used in practically all modern models. In our theoretical work, we considered the tractable analysis case of 2-layer ReLU neural networks trained with the WeightNorm layer, and proved for the first time that dynamically-normalized ReLU neural networks converge to a global minimum when trained with gradient descent under sufficient over-parameterization. In our framework we employed the new tool of the Neural Tangent Kernel and observed that normalization layers result in a decomposition of such kernel, corresponding to length and direction updates. In our analysis, we derived a proof of convergence for two layer neural networks, yet it would be interesting to extend this work to networks with arbitrary number of layers. One such analysis based on the NTK for un-normalized networks is the work of [DLL19a] which presents a lot of the tools to extend the dynamically normalized analysis to the deep settings. On a general note, it would be interesting to see the advancements in practical approaches and theory to finding the fast and general ways of running differentiable programming to optimize and enabling neural networks to “learn” in more flexible environments.

Part II

**A theory for undercompressive shocks
in tears of wine**

Part II presents research in mathematical modelling of thin films of the common tears of wine physical settings. The research presented below is joint work with Hangjie Ji, Claudia Falcon, and Andrea Bertozzi.

CHAPTER 4

Modelling the tears of wine phenomena*

4.1 Introduction

This chapter studies the emergent shock behavior that arises as a result of a solutal Marangoni effect in alcoholic beverages such as wine and Cognac. This scientific project presents a different application of applied mathematics and numerical simulation as compared with Part I of the thesis that fall in the realm of machine learning. Below we give an introduction to the physical problem.

The tears of wine problem is a curious phenomenon that has been observed in wine glasses for centuries. In the right setting, one can observe a thin layer of water-ethanol mixture that travels up inclined surfaces against gravity and proceeds to fall down in the form of tears. This is a result of a solutal Marangoni stress counterbalanced with the force of gravity. The Marangoni stress stems from a surface tension gradient caused by alcohol evaporation and the resulting difference in alcohol concentration. Specifically, when a water-ethanol mixture is placed in a container with inclined walls, a thin meniscus forms. The alcohol in the meniscus region becomes more depleted than that of the bulk due to the predominant role of evaporation in the meniscus. This leads to a solutal surface tension gradient that pulls liquid out of the meniscus and up the side of the glass.

The phenomenon of wine tears was first analyzed qualitatively by Thomson [Tho55] who attributed it to the Marangoni stress. In 1992 the first careful experiments were conducted by

*This chapter is adapted from [DJF20]

Fournier and Cazabat to understand the phenomenon [FC92]. Further studies focused on the various instabilities that form [VEN95, HB01, FC98]. In particular the work of Vuilleumier *et al.* [VEN95] focuses on the stationary state when the film reaches its terminal height and star instabilities form in addition to the tears. In the paper by Fanton and Cazabat [FC98] studying spreading instabilities, the authors continue describing the star instabilities that form in two component mixtures. In 2001, Hosoi and Bush [HB01] further investigated two distinct instabilities in the climbing film using a lubrication model that includes gravity, capillarity and Marangoni stresses. The work of Venerus and Simavilla [VS15] identifies a previously overlooked temperature gradient due to evaporation, that also contributes to the Marangoni stress. More recently, Nikolov *et al.*[NWL18] also applied the Plateau-Rayleigh-Taylor theory to study the ridge instability that triggers the formation of wine tears. However the dynamic formation of the ridge is still not well-understood.

All of the prior works on tears of wine neglected to consider the tangential component of gravity along with the other physics. The tangential Marangoni stress, tangential component of gravity, and the bulk surface tension lead to a dynamic model that is known to produce unusual behavior sometimes characterized by nonclassical shocks. This has been well-studied in thermally driven films [ME06, MB99, Mun03, BMS99, BMF98] but never in tears of wine. In models studied in the literature [HB01, VEN95, VS15], one expects a moving front with advancing fingers, which is inconsistent with the draining tears observed in experiments. This suggests that a more intricate mechanism is taking place, motivating further studies.

Via an enhanced model, we illustrate the existence of nonclassical undercompressive shocks for the first time in the context of tears of wine. This model better characterizes the dynamics of climbing films which sheds light on the experimental work in the literature. Relevant to our analysis are the works studying the structure and shock formation in thermally-driven thin films where nonclassical shocks have been observed [ME06, MB99, BMS99, BMF98]. In this part of the thesis, we investigate different shock morphologies that can spontaneously occur in climbing films of wine, depending on the experimental settings and alcohol concentration.

For instance, we expect undercompressive shocks in the experiments of [VEN95].

More importantly, we take a closer look at the common wine glass setting, something not well-studied in prior works. This corresponds to using a radially symmetric glass, and incorporating swirling as done in common handling of wine. We find that the geometry and swirling of the glass affect the formation of tears, which differs from the better-studied spontaneous climb. Mathematically, the new setting translates to extending the model to incorporate additional geometries, and adding a pre-swirling draining fluid layer. Specifically, our analysis shows that the draining fluid can give rise to reverse undercompressive shocks [Mun03] that help explain the formation of tears from a climbing reverse front, which we find to be quite reproducible, experimentally, with steeper beverage glasses and higher alcohol concentrations.

The rest of Chapter 4 is structured as follows: In section 4.2 we lay out the theory, deriving the non-dimensional PDE model for the climbing thin film. The shape of the meniscus and front dynamics, in addition to the relevant works on the mathematical theory of undercompressive shocks in thin films are introduced in Section 4.3. In Section 4.4 we review the experimental work in the literature, and present numerical simulations of our new model using corresponding experimental parameters. The effects of glass geometries on the film dynamics are investigated in Section 4.5. The appearance of an unusual reverse undercompressive shock wave triggered by a draining film is discussed in Section 4.6. Lastly we discuss our findings on different shocks and hypothesize their relation to the formation of tears in Section 4.7.

4.2 Hydrodynamic model

We derive our model building on the foundational model presented in the work of Fournier and Cazabat [FC92]. Based on conservation of mass of the liquid the authors derive the

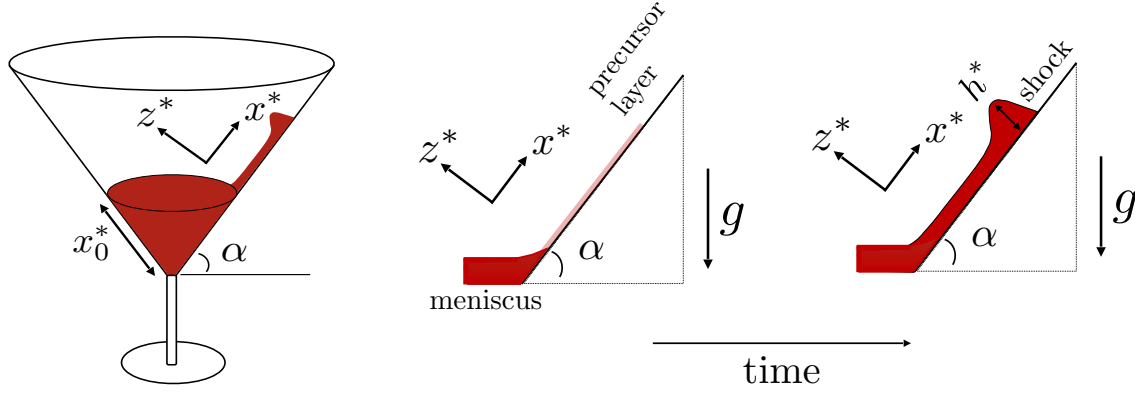


Figure 4.1: (left) Schematic illustration of a conical-shaped cocktail glass of inclination angle α , and (right) the corresponding one-dimensional thin wine film travelling up inclined flat glass surface. The film height h^* is exaggerated for clarity of the illustration.

following equation for the thin film flux

$$\frac{\partial h^*}{\partial t^*} + \frac{\partial Q^*}{\partial x^*} = 0, \quad Q^* = h^* v^*, \quad (4.1)$$

where $h^*(x^*, t^*)$ is the dimensional film thickness, v^* is the average velocity across the film, and Q^* is the flux. Then the velocity is written in terms of the surface tension γ and the dynamic viscosity μ , representing convection of the film due to the surface tension gradient

$$v^* = \frac{h^*}{2\mu} \frac{\partial \gamma}{\partial x^*}, \quad (4.2)$$

We incorporate the tangential and normal components of gravity and the surface tension to the model in equation (4.2) and obtain

$$v^* = \frac{h^*}{2\mu} \frac{\partial \gamma}{\partial x^*} - \frac{h^{*2}}{3} \left(\frac{\rho g_0 \sin \alpha}{\mu} + \frac{\rho g_0 \cos \alpha}{\mu} \frac{\partial h^*}{\partial x^*} - \frac{\gamma}{\mu} \frac{\partial^3 h^*}{\partial x^{*3}} \right), \quad (4.3)$$

where g_0 is gravity, ρ is density, α is the inclination angle of the surface and γ is the surface tension of the film. The formula (4.3) for v^* comes from the lubrication theory [ODB97, FCQ96, CC93, BMS99], which is a long wavelength approximation of the classic Navier-Stokes equations in the low Reynolds number limit. In addition to the first term with surface tension gradient $\partial\gamma/\partial x^*$ from the formula (4.2), the second term in (4.3) represents

the convection of the film due to the component of gravity tangential to the surface, the $\partial h^*/\partial x^*$ term represents the diffusion of the film caused by the normal component of gravity, and the last term with $\partial^3 h^*/\partial x^{*3}$ comes from the surface tension. Using the enhanced model the flux is then reformulated as

$$Q^* = \frac{h^{*2}}{2\mu} \frac{\partial \gamma}{\partial x^*} - \frac{h^{*3}}{3} \left(\frac{\rho g_0 \sin \alpha}{\mu} + \frac{\rho g_0 \cos \alpha}{\mu} \frac{\partial h^*}{\partial x^*} - \frac{\gamma}{\mu} \frac{\partial^3 h^*}{\partial x^{*3}} \right). \quad (4.4)$$

For simplicity we assume a constant surface tension gradient τ following prior works [FC92, VEN95, HB01]. Our model then reduces to

$$\frac{\partial h^*}{\partial t^*} + \frac{\tau}{2\mu} \frac{\partial}{\partial x^*} (h^{*2}) - \frac{\partial}{\partial x^*} \left(\frac{h^{*3}}{3} \frac{g_0 \rho \sin \alpha}{\mu} \right) = \frac{\partial}{\partial x^*} \left[\frac{h^{*3}}{3} \left(\frac{g_0 \rho \cos \alpha}{\mu} \frac{\partial h^*}{\partial x^*} - \frac{\gamma}{\mu} \frac{\partial^3 h^*}{\partial x^{*3}} \right) \right]. \quad (4.5)$$

By balancing the Marangoni stress term and the tangential component of the gravity, we then non-dimensionalize the PDE as in the work of Münch and Evans [ME06] using

$$h^* = Hh, \quad x^* = Xx, \quad t^* = Tt,$$

where

$$H = \frac{3\tau}{2g_0\rho \sin \alpha}, \quad X = \sqrt[3]{\frac{3\gamma\tau}{2(\rho g_0 \sin \alpha)^2}}, \quad T = 2\mu \sqrt[3]{\frac{4\gamma\rho g_0 \sin \alpha}{9\tau^5}}, \quad (4.6)$$

which gives the non-dimensional equation

$$h_t + [f(h)]_x = -(h^3 h_{xxx})_x + D(h^3 h_x)_x. \quad (4.7)$$

Here we denote h, x for the dimensionless film height and length. The constant D is defined as

$$D = \sqrt[3]{\frac{9\tau^2 \cos^3 \alpha}{4\gamma\rho g_0 \sin^4 \alpha}}, \quad (4.8)$$

and the non-convex flux function $f(h)$ takes the form

$$f(h) = h^2 - h^3, \quad (4.9)$$

where the quadratic and cubic terms come from the Marangoni stress and the tangential component of the gravity, respectively. This equation has been studied in thermally driven

h_∞ and b are the left and right boundary conditions of the solution. When analyzing equation (4.7) we consider the travelling wave solutions of the form $h(x, t) = \hat{h}(x - st)$, where s is the speed of the wave. Adjusting to the reference frame of the shock, the flux can be written as $\hat{f}(\hat{h}) = \hat{h}^2 - \hat{h}^3 - s\hat{h}$, which controls \hat{h} via the ODE

$$[\hat{f}(\hat{h})]_x - (\hat{h}^3 \hat{h}_{xxx})_x + D(\hat{h}^3 \hat{h}_x)_x = 0.$$

Integrating this equation using the left and right boundary conditions (4.10) gives the standard Rankine-Hugoniot jump condition for the speed of the shock $s = (f(h_\infty) - f(b)) / (h_\infty - b)$. On the other hand, for large time and space scales one may drop the higher order diffusive terms in equation (4.7) which leads to the quasi-linear hyperbolic equation

$$h_t + [f(h)]_x = 0. \tag{4.11}$$

This reduced equation yields the speed of the characteristics $f'(h) = 2h - 3h^2$. With $\delta = x/t$, PDE (4.11) also admits solutions that consists of an expanding rarefaction wave,

$$h(x, t) = H(\delta), \quad \text{where} \quad H(\delta) = (f')^{-1}(\delta) = \frac{1}{3} - \frac{1}{3}\sqrt{1 - 3\delta}. \tag{4.12}$$

The Lax entropy condition for compressive shocks is given as

$$f'(b) < s < f'(h_\infty), \tag{4.13}$$

or in the moving reference of speed s , $\hat{f}'(b) < 0 < \hat{f}'(h_\infty)$. A characteristic diagram for a compressive shock in the moving reference is illustrated in Figure 4.2 (left) with characteristics entering from both sides of the shock. This type of shock can also be identified as a chord connecting the left and right states of the shock in a flux diagram. One such example is shown in Figure 4.2 (right) where a chord connects the left state $h_{\infty,C} = 0.4$ and the right state $b = 0.1$ of a compressive shock.

Interestingly, for undercompressive shocks the Lax condition (4.13) is violated with

$$f'(b) < f'(h_\infty) < s, \tag{4.14}$$

or in the moving reference of speed s , $\hat{f}'(b) < \hat{f}'(h_\infty) < 0$. This is visualized in Figure 4.2 (middle) where the characteristics travel through the shock, with the undercompressive connection from $h_{\infty,UC} = 0.6$ to $b = 0.1$ plotted in Figure 4.2 (right).

Information propagating through the undercompressive shocks correspond to stability to traverse perturbations [BSB05]. This stability is a mark of undercompressive shocks that does not occur in classical compressive shocks and will be used in distinguishing compressive and undercompressive shocks in the fluid experiments.

Stability of the shock may be analyzed by considering the properties of the perturbed solution, $h(x, t) = \tilde{h}_0(x, t) + \epsilon \tilde{h}_1(x, t) + O(\epsilon^2)$. Here \tilde{h}_0 is a dynamically evolving solution for (4.7) and \tilde{h}_1 is a small perturbation of magnitude $\epsilon \ll 1$. Substituting this ansatz into (4.7) omitting terms of higher order in ϵ , evaluating when the solution is locally constant, and omitting the diffusive terms gives

$$\frac{\partial \tilde{h}_1}{\partial t} + f'(\tilde{h}_0) \frac{\partial \tilde{h}_1}{\partial x} = 0. \quad (4.15)$$

From (4.15) we may deduce the direction that the perturbations travel on either side of the shock. In the frame of the shock we note that the compressive and undercompressive shocks behave differently. For the compressive case (4.13) implies that perturbations will travel into the shock. In contrast, in the undercompressive case (4.14) shows that perturbations travel through the shock. This distinction in perturbation behavior is again illustrated in the characteristic plots in Figure 4.2. As in the undercompressive regime, perturbations travel down and away from the shock, the shock is stable to perturbations unlike the compressive case. We use this criteria as a signature to identify undercompressive shocks emerging from the meniscus.

4.3 Meniscus-driven film climbing and nonclassical shocks

In this section we review prior published experimental results for this problem in which the film climbs onto a dry surface. In this case, the meniscus controls the initial thickness of

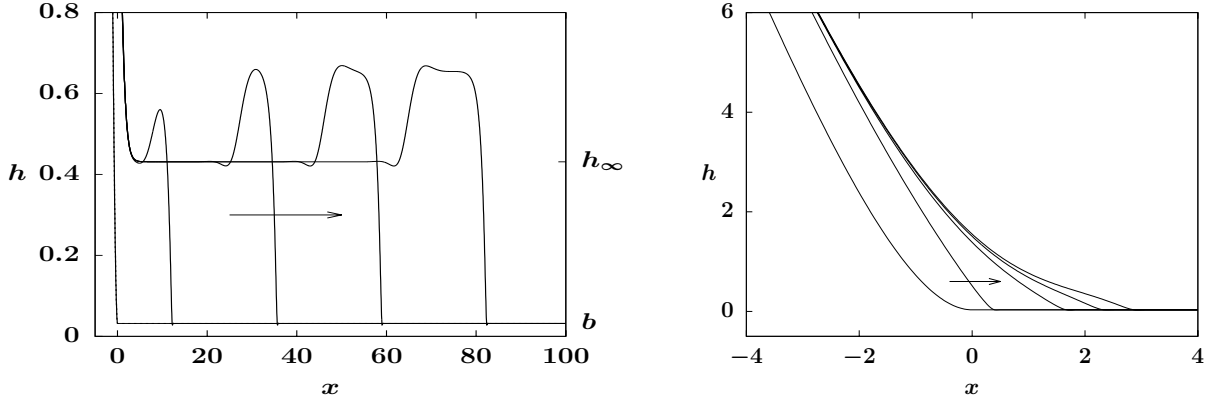


Figure 4.3: (left) Numerical simulation of (4.7) with parameters extracted from experiment (A) [FC92]. The simulation illustrates a compressive-undercompressive shock pair in the long time with $b = 0.0317$ and $h_\infty = 0.43$. (right) Consecutive time steps $\sim 0.1(s)$ starting from the meniscus initial condition.

the film as it climbs, and we can solve equation (4.7) with the meniscus boundary condition, depending on the inclination angle. This same model and boundary condition were already studied in [ME06], however the authors did not consider it in the context of the tears of wine problem.

To model the dynamics of a spontaneous wine film climbing in a static glass, we approximate the boundary condition of equation (4.7) using a meniscus of fixed angle for the left boundary and a precursor pre-wetted layer for the right boundary following [ME06]. The surface of wine in the bulk of the glass is horizontal and meets the thin film at a meniscus angle α . This is expressed as a boundary condition describing the slope of the thin film with the glass, $\partial h / \partial x = \tan \alpha$. In the non-dimensional settings, this gives $\partial h / \partial x = -D^{-1}$, and yields the far-field boundary condition

$$h \rightarrow -x/D \quad \text{for } x \rightarrow -\infty. \quad (4.16a)$$

For the thin precursor layer on the side of the glass we apply the boundary condition,

$$h \rightarrow b \quad \text{for } x \rightarrow \infty, \quad (4.16b)$$

where $b > 0$ is the precursor thickness. The precursor layer on the right boundary is commonly used as a replacement for more complicated contact line models [BB97], and captures the relevant length scale at the contact line. This model alleviates complications that arise with a moving contact line in numerical simulations.

Typical solutions of the PDE (4.7) subject to boundary conditions (4.16) consist of two parts, the meniscus profile and the advancing front (see Figure 4.3 (left)). The meniscus structure is a stationary solution of (4.7) satisfying the far-field boundary condition (4.16a) as $x \rightarrow -\infty$, and selects a flat state of thickness $h_\infty > b$ as the solution advances (see e.g. Figure 4.3). Figure 4.3 (right) shows that a stable meniscus solution is achieved in the numerical simulation of (4.7) starting from the initial data (4.17),

$$h_0(x) = \begin{cases} D^{-3/2}(\exp(D^{1/2}x) - D^{1/2}x - 1) + b & \text{for } x \leq 0, \\ b & \text{for } x > 0. \end{cases} \quad (4.17)$$

This initial condition is a smoothed version of the piecewise linear function that captures the meniscus angle [ME06]. Starting from $h(x, t=0) = h_0(x)$, the simulation uses a standard finite-difference spatial discretization and a backward implicit time-stepping scheme. The spatial derivatives are discretized using upwind scheme with respect to the flux $f(h)$, and central finite-differences for the second and fourth derivative terms.

Away from the meniscus near the apparent moving contact line, the advancing front is given by a traveling wave that connects the left constant state h_∞ and the right thin precursor layer b . Substituting the traveling wave ansatz $h(x, t) = h(\xi)$, $\xi = x - st$ into (4.7), and using the far field boundary condition (4.16b), we get a third-order ODE that determines the advancing front profile

$$-s(h - b) + (f(h) - f(b)) + h^3 h''' - Dh^3 h' = 0, \quad (4.18a)$$

subject to the far-field boundary conditions

$$h \rightarrow h_\infty \text{ for } \xi \rightarrow -\infty, \quad h \rightarrow b \text{ for } \xi \rightarrow \infty, \quad (4.18b)$$

where $' \equiv d/d\xi$. One can have zero, one, or multiple traveling waves depending on the values of the left and right states. This is quite different from the case where the shock is smoothed by ordinary diffusion. Surface tension results in a higher order equation with a complicated solution space [BMS99, ME06].

To match front dynamics with different experiments, one can perform direct PDE simulations of model (4.7) using the meniscus boundary conditions (4.16). For instance, in Figure 4.3 (left), corresponding to the experiment in [FC92], the meniscus dynamics with given $(D, b) = (0.353, 0.0317)$ selects a flat state thickness $h_\infty > b$, and the advancing front consists of two different types of shocks: a compressive shock in the rear and an undercompressive shock at the front of the film. More generally, distinct solution behaviors involving various types of meniscus profiles and advancing fronts can emerge with (D, b) in different parameter regions; this has been extensively studied in [ME06].

Alternatively, for given values of (D, b, h_∞) , one may also use traveling wave solutions satisfying the ODE (4.18) to identify the features of the advancing front [MB99, ME06]. Here the thickness of the left state h_∞ can either be measured experimentally or calculated numerically based on the meniscus dynamics. Instead of revisiting the full dynamics of the meniscus-driven film climbing problem, we briefly review possible shock scenarios characterized by the traveling wave solutions in the context of tears of wine. For a fixed dimensionless $b = 0.0353$, corresponding to the precursor thickness in an experiment from [VEN95], Figure 4.4 (right) summarizes four possible shock scenarios parametrized by h_∞ and D . This bifurcation diagram is numerically obtained by solving the ODE (4.18a) using the asymptotic boundary condition method [GRP01], and is similar to the one studied in [Mun00] for shock transitions in Marangoni gravity-driven thin films.

Four plausible shock structures for (h_∞, D) in different parameter regions are depicted in Figure 4.4 (right): (1) a single compressive shock, (2) a separating double shock pair involving a leading undercompressive wave and a trailing compressive wave (see Figure 4.3 (left)), (3) a rarefaction-undercompressive shock structure, and (4) a generalized Lax shock. The

Shock types	Stability	Figures
Compressive shock	unstable	Figure 4.3 (right)
Compressive-undercompressive double shock	unstable stable	Figure 4.3 (left), Figure 4.5 (left)
Rarefaction-undercompressive shock	stable	Figure 4.5 (right)
Rarefaction-Reverse-undercompressive shock	unstable	Section 4.6

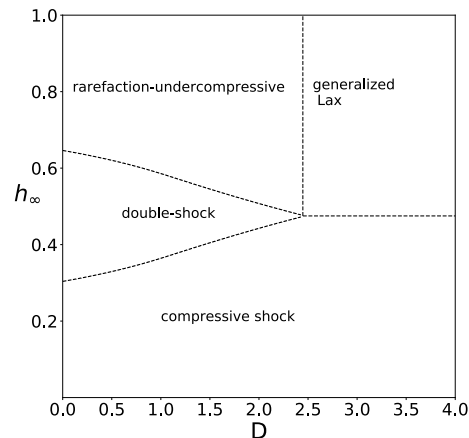


Figure 4.4: (left) Shock types of the hydro-dynamic model (4.7) discussed in the present study; (right) A shock bifurcation diagram parametrized by (h_∞, D) pairs obtained by numerically solving the ODE (4.18a) subject to the boundary condition (4.18b) for $b = 0.0353$.

bifurcation diagram shows that for small values of D , as in most tears of wine experiments from the literature, only shock wave structures of type (1), (2), and (3) can exist. We will discuss these cases using experimental data in the next section. We also present the stability properties of these shocks with respect to transverse perturbations, and point to their corresponding figures in Figure 4.4 (left). In particular, the compressive shock is linearly unstable to transverse perturbations which play an important role in developing later-stage fingering patterns. In contrast, in both the compressive-undercompressive shock pair and the rarefaction-undercompressive shock, the leading undercompressive front is stable and prevents fingering from happening in the contact line [BMS99, BMF98, BSB05].

Another type of shock, reverse-undercompressive shock, is also observed in the study of tears of wine dynamics after a glass swirling. Modified initial and boundary conditions will be used to characterize this scenario. This is documented in Figure 4.4 (left), and we will discuss this case in detail in section 4.6.

4.4 Experimental survey and simulations

Now that we have a nonlinear model for the wetting behavior of the climbing film, we can compare it with experimental data in the prior literature. However, the behavior of undercompressive shocks depends very sensitively on the dimensionless parameter b . Very few experiments study this in detail - one example being [SC00] for thermally driven films which are easier to control. Likewise τ can sometimes be hard to measure and it appears in the calculation of both b and D , which are the dimensionless parameters needed to model the experimental data. We analyze the effect of the uncertainty of these parameters here.

We consider the prior works: (A) the seminal “Tears of Wine” [FC92] paper and (B) “Tears of wine: the stationary state” [VEN95]. (A) presents several experiments from which we use the parameters corresponding to alcohol concentration $C = 70\%$. This experiment has the most detailed measurements and also shares some measurements with Vuiellemuier *et al.*[VEN95]. For (B) we analyze two physical experiments: Experiment I, that follows the experimental settings of Figure 5b of their paper with a curvature-driven film and $C = 70\%$, and experiment II that refers to Figure 5b of (B) and follows a gravity-driven regime with $C = 70\%$. We label the experiments as (BI) and (BII) and note that they correspond to the same physical setting with different assumptions when deriving the surface tension gradient τ .

In Appendix 4.A we provide a complete set of measurements for each experiment, as well as the dimensionless values (D, b) needed for analysis. Using different (D, b) values corresponding to each experiment we conduct a sequence of numerical simulations for equation (4.7). The initial and boundary conditions are specified as in (4.16 – 4.17). In Figure 4.5 we present numerical simulations for (BI) and (BII) and observe that despite identical physical settings the different values of τ lead to different shocks. In particular (BI) exhibits a compressive-undercompressive shock while (BII) has a single undercompressive shock front.

In addition to τ , the precursor thickness b is also of key importance to the dynamics of the advancing front [BMF98]. For example for the setting of (A) that leads to an advancing

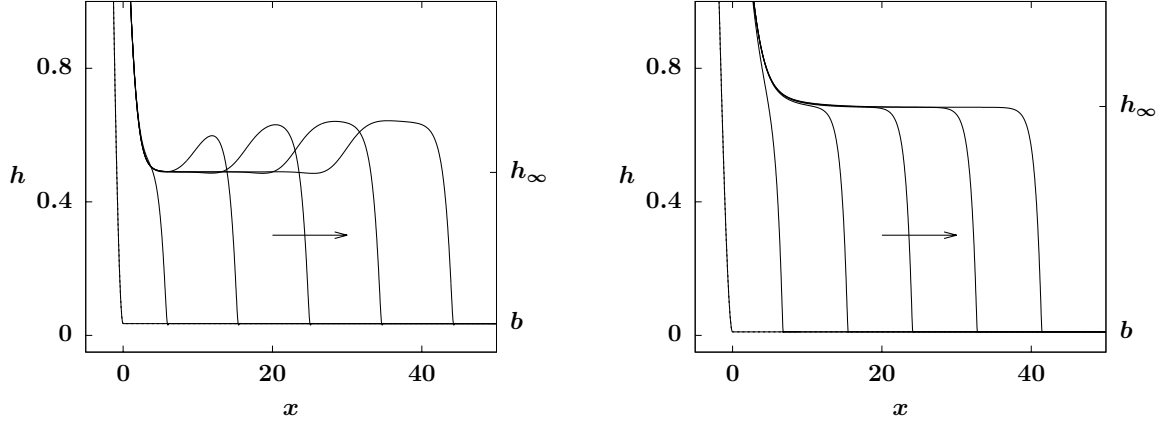


Figure 4.5: Numerical simulations of Vuilleminier *et al.* experiments (BI), (BII) for long times. (left) experiment (BI) exhibiting a less distinct compressive-undercompressive double shock cf. Figure 4.3. (right) experiment (BII) exhibiting undercompressive shock.

front with a compressive–undercompressive double shock, when the precursor thickness is increased from $b = 0.0353$ to $b = 0.1585$ the front transitions into a single compressive wave. We observe this change in behavior while the rest of the settings are fixed (see Figure 4.6).

For other experiments in the literature [HB01, VS15] (see Appendix 4.A for complete listing) the authors did not report their data for the climb of the film. Therefore we cannot fully compare our theory against their experimental observations. In our experiments that match the high inclination angle and ethanol-water fraction of [HB01, VS15], we do not observe easily reproducible film climbing.

With a hypothetical thin precursor thickness, our simulations of [HB01, VS15] based on the meniscus-driven film dynamics predict a thin compressive advancing front. In Figure 4.7 we present the evolution of a thin film climbing out of the meniscus using the dimensionless parameter $D = 0.0338$ that corresponds to Table 3, Figure 10 of [HB01]. A small precursor thickness $b = 0.028$ (corresponding to a dimensional thickness of $b^* = 0.5\mu\text{m}$) is used to approximate the dry substrate. For our experiment, in Figure 4.8, we show a spontaneous climb that is similar to experiment (A). The settings of our experiments are of a watch glass of diameter 75 mm and angle $9^\circ < \alpha < 20^\circ$ (due to curvature of the watch glass). For this high

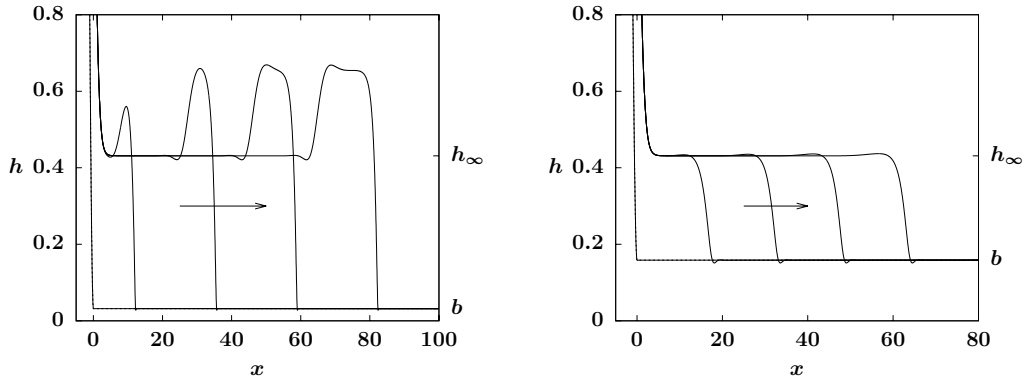


Figure 4.6: A comparison of shock types affected by the precursor thickness b , showing that (left) $b = 0.0353$ results in a compressive-undercompressive shock, and (right) $b = 0.1585$ (corresponding to a hypothetical dimensional thickness $b^* = 10\mu\text{m}$) leads to the formation of a compressive wave. The other parameters in the two simulations are identical to those in Figure 4.3 and correspond to measurements from experiment (A).

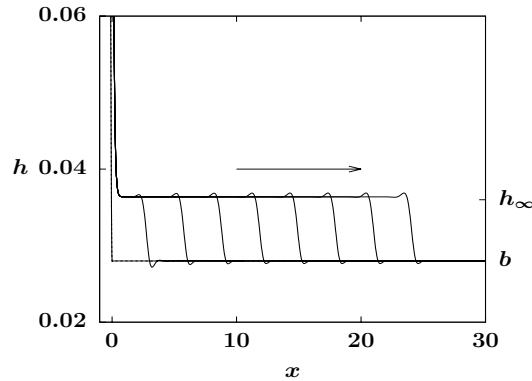


Figure 4.7: The evolution of film height of [VS15] (wine setting) with $b = 0.028$ (corresponding to a hypothetical dimensional thickness $b^* = 0.5\mu\text{m}$) showing the formation of a compressive wave. The meniscus dynamics with $D = 0.0338$ yields a left state thickness $h_\infty = 0.036$.

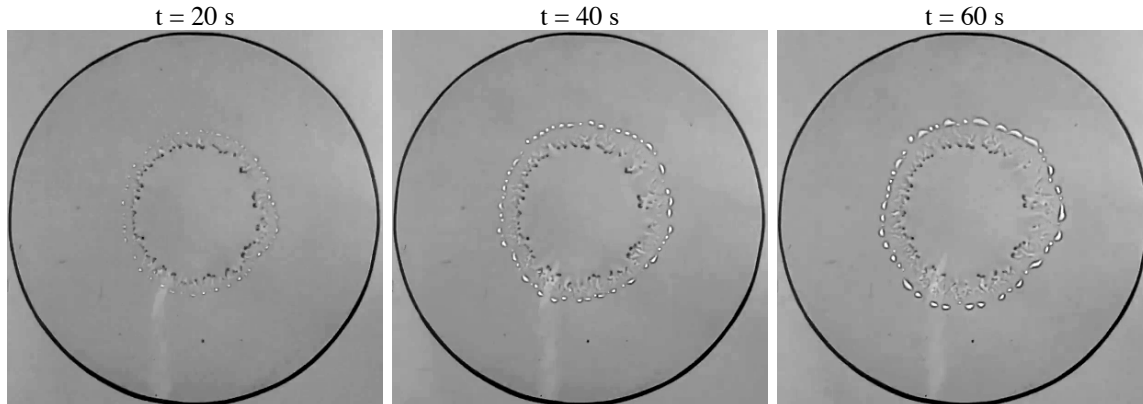


Figure 4.8: Spontaneous climb images of ethanol-water mixture with ethanol concentration $C = 0.7$ on a dry watch glass (no pre-swirl) of diameter 75 mm and angle ranging between $9^\circ < \alpha < 20^\circ$.

alcohol concentration and inclination, the climbing film on the dry substrate does exhibit a spontaneous climb.

In our model, different settings lead to different shock structures. This is in contrast with the previous literature, where a model with a surface tension gradient and tangential gravity is used and only a single type of shock emerges. Without the competition between the surface tension gradient τ and the tangential component of gravity, we only observe classical compressive shocks. When incorporating both gravity and surface tension as in (4.7), different physical parameters (in particular the substrate wetting thickness b) lead to qualitatively different shocks. In Appendix 4.A we present tables of the prior works with some photographs from the experiments. More work is needed to better understand quantitatively how shocks behave in tears of wine on a dry surface. Going forward here, we show that in the case of a surface coated by swirling, one can obtain very reproducible shock profiles, which our theory suggests are reverse undercompressive shocks. We distinguish pre-swirling from the precursor discussed in Sections 4.2 – 4.5, noting that in the preswirled regime, the right boundary condition maintains a thicker fluid film. This is further discussed in section 4.6. In the next section we present a model for a conical shaped substrate (as in our experiments)

rather than a flat substrate. We show that it results in minor modifications to the behavior. The flat surface case is important because the model reduces to a regular scalar conservation law for which there is a well-developed shock theory.

4.5 Conical shaped substrate

So far we have assumed negligible curvature effects of the substrate. In this section we investigate the substrate curvature effects on shock formation. For simplicity, we consider an axisymmetric thin fluid film climbing up the surface of a conical-shaped cocktail glass of inclination angle α (see Figure 4.1 (left)). In the long-wave limit the balance of normal stresses at the free surface $z = h(x, t)$ yields the leading-order equation

$$p = \frac{A}{x + x_0} - \frac{\partial^2 h}{\partial x^2}, \quad (4.19)$$

where p is the dynamic pressure, the term $A/(x + x_0)$ represents the azimuthal curvature of the conical substrate, $x_0 > 0$ measures the distance between the surface of the wine reservoir/bulk and the vertex of the cone, and the non-dimensional parameter A is given by $A = X/(H \cot \alpha)$ for the length-scales X and H defined in (4.6). Using a PDE derived in [RRS02] and studied in [GBS06] for the dynamics of thin films driven by gravity and surface tension on a curved substrate, we write the non-dimensional governing equation for the film thickness $h(x, t)$ as

$$\frac{\partial \zeta}{\partial t} + \frac{\partial}{\partial x} (h^2 - h^2 \zeta) = \frac{\partial}{\partial x} \left(h^2 \zeta \frac{\partial p}{\partial x} \right) + D \frac{\partial}{\partial x} \left(h^3 \frac{\partial h}{\partial x} \right), \quad x \geq 0, \quad (4.20)$$

where ζ represents the amount of fluid above a surface patch and is approximated by

$$\zeta = h - \frac{\kappa h^2}{x + x_0}, \quad (4.21)$$

where the non-dimensional quantity $\kappa = H/(2X \cot \alpha)$ arises from the principle curvature of the substrate. This model characterizes the joint effects of substrate curvature, constant surface tension gradient, and both normal and tangential components of the gravity. For typical tears of wine experiments we have $H/X \ll 1$ and $\kappa \ll 1$, therefore we approximate ζ

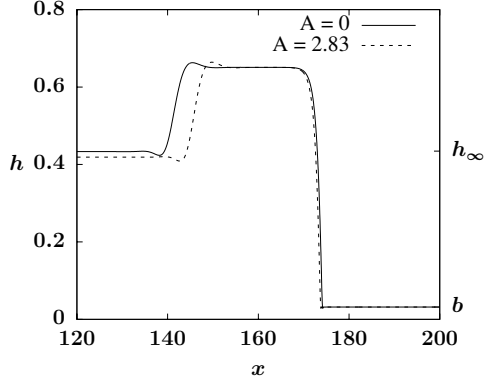


Figure 4.9: Long-time shock profiles of (4.22) for $A = 0$ (no curvature effects) and $A = 2.83$, $x_0 = 5$ (with curvature effects) at $t = 700$. Other system parameters are $b = 0.0317$, $D = 0.353$ corresponding to experiment (A).

by h , and rewrite equation (4.20) using (4.19) by

$$h_t + (h^2 - h^3)_x = - \left[h^3 \left(-\frac{A}{(x + x_0)^2} + h_{xxx} \right) \right]_x + D (h^3 h_x)_x. \quad (4.22)$$

Here it is important to have $x_0 > 0$ to avoid the shape singularity at the vertex. In the limit $x \rightarrow \infty$, the azimuthal curvature term is dropped and the model (4.22) reduces to (4.7).

Using experimental parameters in experiment (A) with a small inclination angle $\alpha = 9^\circ$, we plot in Figure 4.9 the comparison of long-time shock profiles without curvature effects ($A = 0$) and with finite curvature effects ($A = 2.83$, $x_0 = 5$). It shows that incorporating the substrate curvature effects lowers the thickness of the left constant state h_∞ , and makes the separation of the leading undercompressive wave and the trailing compressive wave in the double shock pair less pronounced. Based on the theory for shock transitions in model (4.7) (or equivalently (4.22) with $A = 0$) developed in [BMS99, Mun00], for fixed D and b values, decreasing the value of h_∞ can push the solution out of the double shock regime and into the compressive regime (see Figure 4.4 (right)). This is consistent with our observation in Figure 4.9 with finite curvature effects ($A > 0$), where the less pronounced separation of fronts caused by the decreased h_∞ suggests a transition to compressive waves.

While the model (4.22) is limited to the dynamics on a conical shaped substrate, a

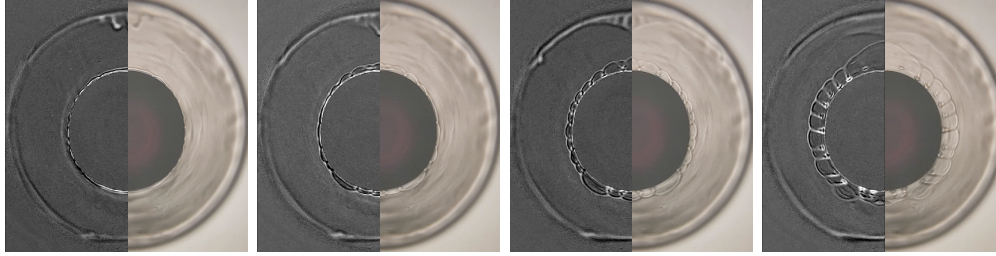


Figure 4.10: Top view images of tears of wine experiment at $t = 0, 5, 10, 20$ s in a stemless Martini glass (conical substrate) using 18% alcohol by volume Port wine. Swirling the wine around the glass creates a reverse front that forms out of the meniscus, advances up the glass and destabilizes into wine tears.

generalized nonlinear model incorporating the substrate geometry of wine glasses can be obtained by using a different functional term for the azimuthal curvature term. More complicated curvature-induced shock transitions are expected to occur, and we refer the readers to the work of Roys *et al.*[RRS02] and Greer *et al.*[GBS06] for a detailed discussion of the modeling and numerical methods of lubrication models on a curved substrate.

4.6 Reverse undercompressive shocks on a preswirled substrate

It is difficult to reproduce many of the experiments performed with an initially dry surface discussed in the prior literature, because of the need to control the wetting properties of the contact line. Ordinary glassware will be affected by the way it is cleaned (e.g. see [App15] in which the author claims that glasses cleaned in a dishwasher with an additive to avoid spotting makes it more difficult to see wine tears). However, one can quickly observe wine tears by actively pre-wetting (pre-swirling) the glass as one would do when drinking a beverage or swirling the wine in the glass before drinking it. A preswirled glass can produce dramatic wine tears (see [Dan15] for an illustration). For the first time in the context of tears of wine, we identify the existence of another fundamental type of shock, the reverse undercompressive (RUC) shock, that involves a thicker film receding from a thinning region.

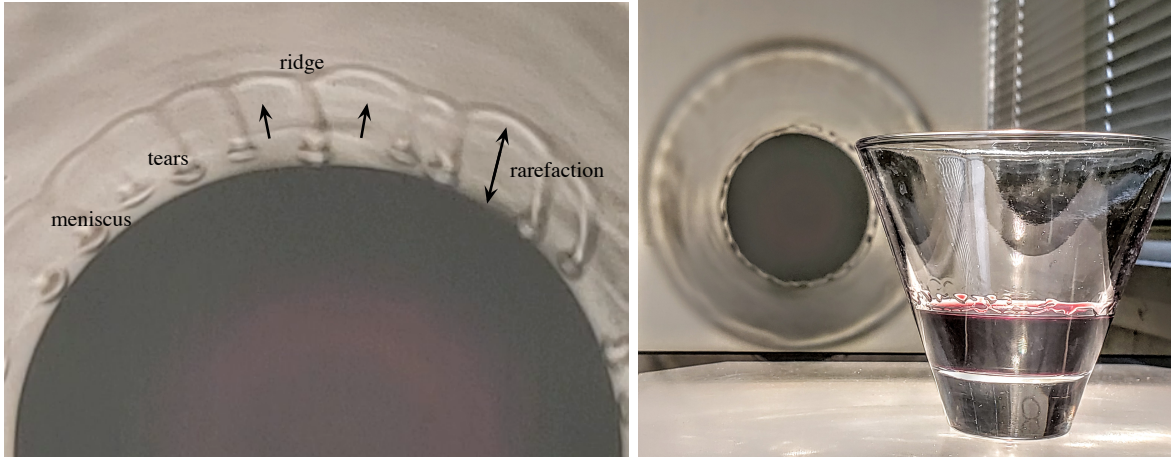


Figure 4.11: Tears of wine experiment. (left) top view and (right) side view and projection of a stemless martini glass with inclination angle $\alpha = 65^\circ$, using 18% Port wine. Swirling the wine around the glass creates a front that forms out of the meniscus. The draining film advances up the glass and destabilizes into wine tears.

Thin film structures involving an undercompressive leading shock and a trailing RUC shock were first identified in [Mun03, SBB03] for dip-coating experiments with a thermal gradient that drives the film against gravity. The model used in [Mun03, SBB03] is the same one we consider here.

Our experiments are performed using port wine of alcohol concentration $C = 18\%$ and a stemless martini glass of inclination angle $\alpha = 65^\circ$ in a room with controlled temperature at 75°F . One can cover the glass immediately after pouring the wine, to temporarily suppress the evaporation of alcohol. A few seconds after pouring and covering, we give the covered glass a brief slow swirl for about 3 seconds and coat the substrate. We observe that the initial swirl provides a surface with a thin draining film. We leave the cover on for ~ 10 seconds until the swirl is no longer visible and the draining has settled down. After removing the cover, evaporation quickly increases, inciting a “reverse” front to climb out of the meniscus, followed by the formation of wine tears falling back into the bulk. The experiments are highly reproducible and the times indicated here, as long as they are in the order of seconds, for swirling, covering, and uncovering, do not affect the outcome observed. This is supported

by the theory for a range of preswirled thickness. The forming front is characterized by a depression, i.e. the film ahead of the front is thicker than the film behind it. It is in a sense, a “dewetting” front that leaves a thinner layer behind it. The formation of the moving front is initiated by a pinch-off that occurs in the meniscus, as predicted in [Mun03]. Snapshots of this experiment displayed in Figure 4.10 show a front that appears out of the meniscus and destabilizes into wine tears after ~ 10 seconds. The left half of each image is a reflection, that is enhanced to visualize and capture the moving front. Around the center we observe a circular wave forming and travelling outward from the meniscus up the glass. The tears originate from the instability of that wave and drain back into the bulk fluid. Such waves appear to be the dominant behavior in the formation of the actual “tears of wine”. We note that the “swirling” initial condition may lead to different film thicknesses depending on the force of the swirl, i.e. the coating thickness is not quantitatively reproducible by manual swirling. We now present a theory that shows that such film thicknesses, within a fairly broad range, all produce the same general pattern of a reverse undercompressive wave emerging from the meniscus, as in Figure 4.11. The predicted front behavior is universal within a range of coating thicknesses, to the point where one can do reproducible demonstrations at the dinner table. Another signature that this is an RUC shock is that the tears emanate from the wave and travel downward, away from the shock, and towards the meniscus. This is indicative of characteristics going through the wave, away from its direction of travel, because perturbations, to leading order, travel along characteristics. A diagram for this type of behavior is shown in Figure 4.2 in the middle panel. This is in contrast to a compressive wave in which disturbances, traveling along characteristics, enter the shock from both sides. One would expect instabilities of a compressive wave to travel with the wave, like in the case of the fingering instabilities seen in Figure 4.8.

We now match the observed experimental behavior in Figure 4.11 to solutions of the one dimensional model. To approximate the initial profile of the draining film, after swirling, and immediately after the evaporation starts, we assume a pinch-off of the meniscus, as discussed in [Mun03]. For simplicity we start with an initial condition that has steep constant slope

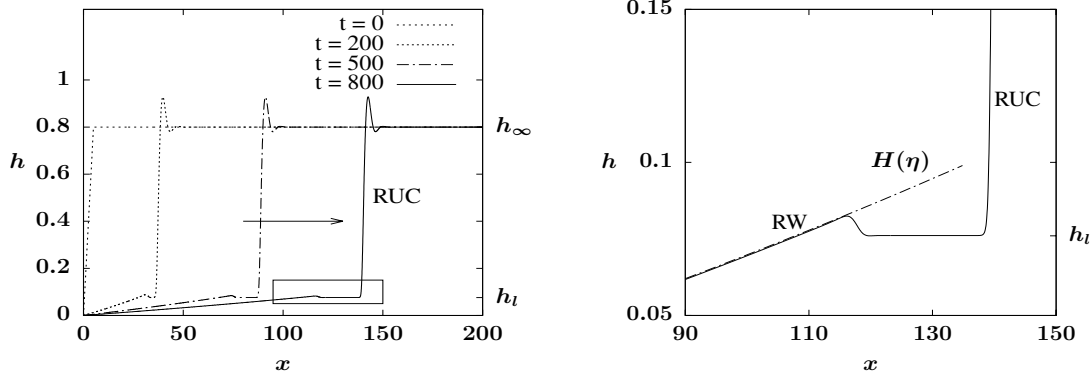


Figure 4.12: (left) The formation of a reverse–undercompressive (RUC) shock with initial condition (4.23) and boundary conditions (4.24). (right) A closeup view of the boxed region on the left, at dimensionless time $t = 800$, showing a good agreement with the rarefaction wave (dot-dashed line) $H(\delta)$ in (4.12) for $\delta = x/t$. The rarefaction separating from the RUC shock is a signature of a nonclassical shock, it is shown by the flat state h_l in between the rarefaction and the RUC shock.

jump connecting the meniscus to the coating layer,

$$h_0(x) = \begin{cases} h_\infty & \text{for } x > x_L, \\ h_{eq} + \frac{h_\infty - h_{eq}}{x_L} x & \text{for } 0 < x \leq x_L, \end{cases} \quad (4.23)$$

where the film thickness h_{eq} approximates the near-rupture film profile near the edge of the meniscus (as in Figure 6 from [VEN95]), and h_∞ sets the thickness of the draining film due to the swirling of the glass. We take h_∞ to be independent of time however a more complete model could include a weak time dependence due to the dynamics further up the glass. For the lower boundary conditions we apply

$$h(0) = h_{eq}, \quad h_{xxx}(0) = 0, \quad (4.24a)$$

which assumes a fixed near-rupture film thickness and a zero curvature gradient at $x = 0$. The upper boundary condition is

$$h \rightarrow h_\infty \quad \text{for } x \rightarrow \infty. \quad (4.24b)$$

Our numerical simulations of equation (4.7) have $D = 0.0146$, $x_L = 5$, and $h_{eq} = 0.001$. These dimensionless parameters correspond to the dimensional values taken from [VS15] (See in Appendix 4.A experiment (DI) of Table 4.2) with a modified inclination angle $\alpha = 65^\circ$. Unlike the cases discussed in Section 4.4 where the film thickness h_∞ is determined by the parameters D and b based on the meniscus dynamics, here we specify the value of h_∞ to approximate the thickness of the initial draining film formed by the glass swirling in the experiment. We will pick typical values for h_∞ that correspond to a balance between the draining effect and the Marangoni stress. We find that a wide range of such h_∞ produce the same qualitative behavior.

Figure 4.12 (left) shows a typical numerical simulation of the model (4.7) for the evolution of the film height starting from the initial condition (4.23) with $h_\infty = 0.8$. The left-hand boundary models the pinchoff at the meniscus, a phenomenon that has been widely studied in coating films [Mun03, CC93], and is driven by the dynamics of the meniscus as it forms the equilibrium height h_{eq} . This pinchoff leads to a pronounced complex wave form emanating from the meniscus. In the early stage of the dynamics, we see a double wave structure emerging. There is a rarefaction fan near the meniscus and a shock wave connecting to the larger height h_∞ . Note that the two waves separate from each other as time passes. A flat film of thickness $h = h_l$ (see the tick labels on the right vertical axes in Figure 4.12) connects the right edge of the rarefaction wave and the left edge of the leading wave. This is typical for such double wave structures involving undercompressive waves — a new equilibrium height emerges that is driven by the solution on each side [Mun03, BMS99]. Figure 4.12 (right) shows a close-up of the solution profile at $t = 800$ delimited by a box in Figure 4.12 (left), indicating that the rarefaction wave (RW) portion of the solution is given by $h(x, t) = H(x/t)$ in (4.12). To further verify the UC structure, we plot the connection between h_l and h_∞ on the flux function diagram (see Figure 4.13, right panel). The chord crosses the graph of the flux function, illustrating that the shock violates the entropy condition.

Figure 4.13 shows film heights with $h_\infty = 0.4, 0.6, 0.8$. For $h_\infty = 0.8$, we have $h_l \approx 0.076$;

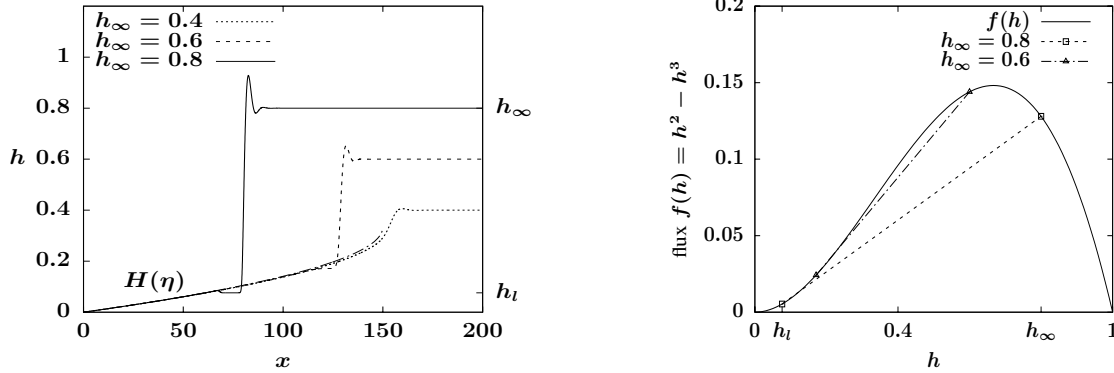


Figure 4.13: (left) A comparison of shock solutions at $t = 450$ for initial data (4.23) with varying h_∞ showing reverse-undercompressive (RUC) shocks for $h_\infty = 0.6, 0.8$, and a single rarefaction wave for $h_\infty = 0.4$. The critical thicknesses h_∞ and h_l are marked for the $h_\infty = 0.8$ profile. Note that the rarefaction part of the solution is independent of h_∞ . (right) The flux diagram with two undercompressive connections for the RUC shocks with $h_\infty = 0.6, 0.8$.

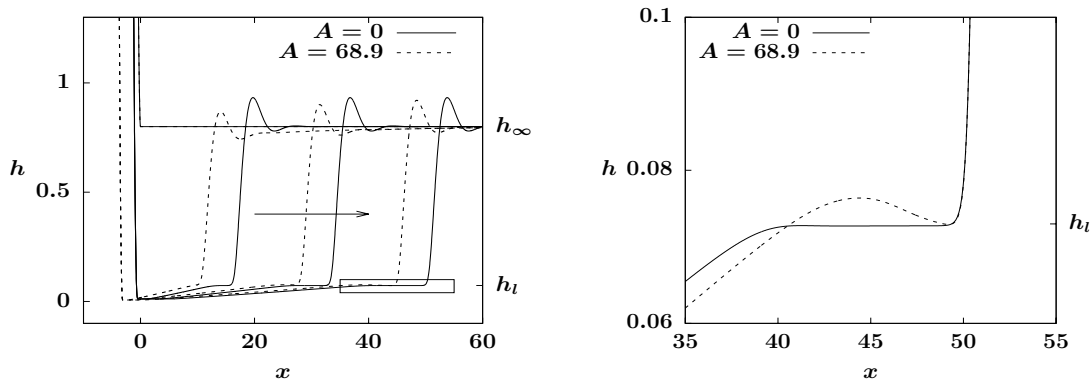


Figure 4.14: (left) Advancing waves starting from initial condition (4.17) at times $t = 0, 100, 200, 300$ governed by (4.22) with meniscus boundary conditions (4.16), showing a comparison of fronts influenced by curvature effects ($A = 68.9, x_0 = 25$) and without curvature effects ($A = 0$). (right) A closeup view of the boxed region on the left at time $t = 300$ with the curve for $A = 68.9$ shifted by $\Delta x = 5.3$. The other settings are identical to those in Figure 4.12.

for $h_\infty = 0.6$, $h_l \approx 0.17$. In both cases the shock violates the entropy condition. For $h_\infty = 0.4$ the dynamics is dominated by the rarefaction fan which terminates abruptly in the flat film on the right hand side without the pronounced capillary ridge seen in the other two cases. In this case, the dynamics are dominated by the Marangoni stress. For really thick draining films (e.g. with $h_\infty > 1$), the dynamics is dominated by gravity so we expect a range of h_∞ for which this phenomenon occurs. Incorporating the conical-shaped substrate curvature effects and the meniscus dynamics also influences the profile of the RUC shock. Here we combine these effects by using model (4.22) with the meniscus boundary conditions (4.16). Starting from the initial condition (4.17) that emulates the meniscus profile, in Figure 4.14 (left) we plot the simulation results with $A = 68.9$, $x_0 = 25$ against the profiles without curvature effects ($A = 0$). Other system parameters are set to be $(D, h_\infty) = (0.0146, 0.8)$ which match the simulation shown in Figure 4.12. This comparison shows that the early stage pinch-off near the meniscus is sensitive to the substrate curvature effects, which leads to a different stable meniscus profile and location where the near-rupture film thickness h_{eq} is attained. This difference leads to a spatial shift in the later stage dynamics, whereas the rarefaction wave and the speed of the moving front do not change significantly. A closeup view of the wave fronts at $t = 300$ is also shown in Figure 4.14 (right), where the curve for $A = 68.9$ is horizontally shifted to align with the $A = 0$ curve. It indicates that the RUC shock obtained for $A = 0$ is less pronounced with the presence of weak substrate curvature effects.

Previously, it has been shown that the RUC wave is unstable with respect to transverse perturbations [Mun03]. As the wave destabilizes, the transverse perturbations enter the space between the RUC wave and the rarefaction fan, which agrees with the wine tears being shed downward from the rising circular wave in our experiment (see Figure 4.11). As time goes on, the tears travel into the rarefaction fan and get elongated as the rarefaction wave expands. The theory here suggests a mechanism for the onset of the wine tears. A fully nonlinear 2D simulation of the model could be done in future work to understand the longer time dynamics of the wine tears.

4.7 Conclusion

In Part II of the thesis, we introduce a model for the tears of wine phenomena that describes the balance between gravity and a Marangoni stress induced from alcohol evaporation. The dynamic model is the same equation that has been used to describe thermally driven films balanced by gravity. This work is the first to connect that literature to the tears of wine problem. We argue that the actual wine tears, which drain down the glass, in contrast to the well-known fingering instability of driven fronts, which travel in the same direction of the front, arise from an instability of a reverse undercompressive shock. They can be easily observed by prewetting the glass as one would do in the context of drinking a beverage or swirling the wine around the glass. We are able to create fairly reproducible experiments of this phenomenon by pre-swirling the glass, while covered, to suppress evaporation. Removing the cover, after the initial pre-swirl, leads to a circular wave emanating from the meniscus that quickly destabilizes into downward draining wine tears.

Our main model is for a flat substrate. This model allows for easy identification of different wave forms because they have an exact self-similar structure. We also show that incorporating the substrate curvature effects into the governing equation can lead to dynamic behaviors that are qualitatively similar, and the difference can be quantified through numerical simulations.

It has been shown in the literature [BMS99, BMF98] that while the undercompressive shocks are stable, the compressive shocks and reverse undercompressive shocks are unstable to fingering [Mun03]. More work could be done to quantitatively predict the spacing of the wine tears observed in these experiments. This would involve analyzing the linear stability of the RUC ridge along with fully nonlinear 2D numerical simulations.

Prior experimental results presented in Section 4.4 illustrate the formation of different shock structures under different experimental conditions. For example we observe that the surface tension gradient τ and the precursor height b are pivotal to the formations of different shocks. While a conical-shaped martini glass is easy to model because of its constant inclination angle, one could also incorporate three-dimensional complex surface

geometry to the model such as that observed in common wine glasses. We believe a more accurate description of the phenomenon may be obtained via a careful consideration of the three-dimensional geometry and the surface tension gradient. Finally, we note that our model (4.7) assumes a constant surface tension gradient. As the film climbs up, this assumption eventually fails, requiring a modification of the model to describe the full dynamics. Along these lines, downward draining wine tears can also be observed from fluid that accumulates at the top of the glass, forming a stationary capillary ridge [COM15, NWL18, VS15]. As a future work, it would be insightful to investigate a model for the tear formation structure.

4.A Appendix: Extended survey of prior experimental works

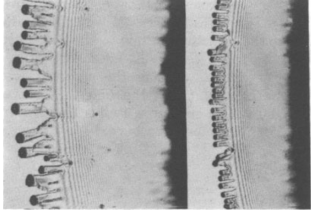
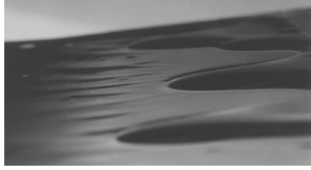
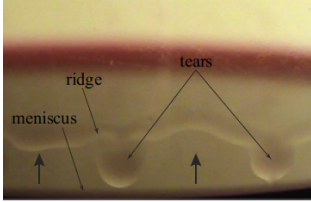
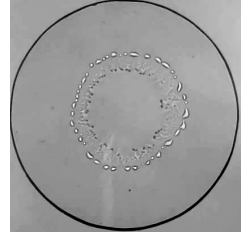
Here we review the existing experimental literature summarized in Table 4.2 and 4.3. The works discussed are: (A) “Tears of Wine” [FC92], with alcohol concentration $C = 70\%$, (B) “Tears of wine: the stationary state” [VEN95] with experiments (BI) of data taken from Figure 5b of their paper with a curvature-driven film and $C = 70\%$. and (BII) of data taken from Figure 5b, now with a gravity-driven regime and $C = 70\%$. As mentioned in section 4.4, (BI) and (BII) refer to the same physical setting with different assumptions when calculating the surface tension gradient τ . (C) “Evaporative Instabilities in Thin Films” [HB01] experiment (CI) refers to the settings described in Table 3 and Figure 10 of [HB01]. Experiment (CII) refers to Table 3 in [HB01] but in addition uses the experimental settings given in Figure 11 of the same paper. (D) “Tears of wine: new theory on an old phenomena” [VS15] presents two experiments, for wine and cognac. We denote the experiments as (DI) and (DII). In the tables, b^* refers to the precursor thin film height measured in μm , h_∞^* (μm) refers to the height of the film at the bulk (of the thin film), γ refers to the surface tension (N/m), τ (Pa) refers to the surface tension gradient, α is the inclination angle measured in degrees, μ is the dynamic viscosity of the film (mili- $Pa\ s$), and C is the volumetric water-ethanol fraction. The collection of symbols and typical dimensional values are also presented in Table 4.1 for convenience.

In the third column of the tables of the experiments we present dimensionless values for (D, b) that appear in the PDE model in equation (4.7). We remark that some of the values we present are interpolated from other experiments. For example, in Table 4.2, the dimensional precursor value b^* is only provided in the literature for experiment (A). For simplicity, we use the dimensional precursor thickness $b^* = 2\mu m$ of (A) for the other experiments with high alcohol concentrations and low inclination angles (experiments, (BI), (BII), and (CI)). For experiments with higher inclination angles and lower alcohol concentrations (experiments (CII), (DI), and (DII)), the authors did not report the precursor height and we may not interpolate it since we do not have b^* measurements for such settings.

Table 4.1: Relevant dimensional groups used in Table 4.2

Physical Quantity	Symbol	Typical dim. value	Dimensionless range
Upstream thickness	h_{∞}^*, h_{∞}	$30\mu m - 98\mu m$	0–2.1
Precursor thickness	b^*, b	$2\mu m$	$10^{-2} - 10^{-1}$
Surface tension	γ	$22.39mN/m - 72.86mN/m$	
Surface tension gradient	τ		
Inclination angle	α	$7^{\circ} - 45^{\circ}$	
Fluid dynamic viscosity	μ	$1.1mPa \cdot s^{\S}$	
Alcohol concentration	C	0.15–0.7	
Density	ρ	$784kg/m^3 - 973kg/m^3$	

Table 4.2: Experimental results from literature and corresponding theory

Experiment	Dimensional constants	Dimensionless constants	Images of experimental results
(A) Tears of Wine (Fournier and Cazabat) [FC92]	$h_{\infty}^* = 55\mu m$ $b^* = 2\mu m$ $\gamma = 0.0298N/m^{**}$ $\tau = 0.055Pa$ $\alpha = 9^\circ$ $\mu = 2.1mPa\ s^{\S}$ $C = 0.7$ $\rho = 852\ kg/m^3$	$b = 0.0317$ $D = 0.353$	 Experiment (A) settings at different times
(CI) Evaporative instabilities in climbing films experiment I (Hosoi and Bush) [HB01]	$h_{\infty}^* = 30\mu m$ $b^* = 2\mu m^{\dagger}$ $\gamma = 0.027N/m$ $\tau = 0.025Pa$ $\alpha = 4^\circ$ $\mu = 1mPa\ s^{\S}$ $C = 0.65 - 0.7$ $\rho = 852\ kg/m^3$	$b = 0.031$ $D = 0.639$	 $\alpha = 4^\circ$ Concentration as in (CI) Reprinted with permission from [HB01] and the Cambridge University Press.
(DI) Tears of wine: new insights on an old phenomenon (wine) (Venerus and Simavilla) [VS15]	$\gamma = 0.054N/m^{**}$ $\tau = 0.08Pa$ $\alpha = 45^\circ$ $\mu = 1.1mPa\ s^{\S}$ $C = 0.13$ $\rho = 973\ kg/m^3$	$D = 0.0338$	 Image of experiment (DI) ¶
Our experiment shown in Figure 4.8	$b^* = 2\mu m^{\dagger}$ $\gamma = 0.0298N/m^{**}$ $\tau = 0.055Pa$ $\alpha = 9^\circ$ $\mu = 2.1mPa\ s^{\S}$ $C = 0.7$ $\rho = 852\ kg/m^3$ Set-up matches (A)	$b = 0.0317$ $D = 0.353$	

** refers to surface tension interpolated from [VAN95] § refers to viscosity interpolated from [Vis]

† refers to precursor thickness taken from [FC92] ¶ Reproduced from “Tears of wine: new insights on an old phenomenon”

[VS15] under compliance with the creative commons 4.0 licence.

Table 4.3: Additional experimental results from literature and corresponding theory

Experiment	Dimensional constants	Dimensionless constants
(CII) Evaporative instabilities in climbing films experiment II (Hosoi and Bush) [HB01]	$\gamma = 0.027N/m$ $\tau = 0.025Pa, \alpha = 20.05^\circ$ $\mu = 1mPa \text{ s}^\S, C = 0.65 - 0.7$ $\rho = 852 \text{ kg}/m^3$	$D = 0.072$
(BI) Tear of wine: The stationary state experiment I (Vuilleumier et al.) [VEN95]	$h_\infty^* = 98\mu m, b^* = 2\mu m^\dagger$ $\gamma = 0.0298N/m^{**}, \tau = 0.033Pa$ $\alpha = 6^\circ, \mu = 2.1mPa \text{ s}^\S$ $C = 0.7, \rho = 852 \text{ kg}/m^3$	$b = 0.0353$ $D = 0.43$
(BII) Tear of wine: The stationary state experiment II (Vuilleumier et al.) [VEN95]	$b^* = 2\mu m^\dagger, \gamma = 0.0298N/m^{**}$ $\tau = 0.10Pa, \alpha = 6^\circ$ $\mu = 2.1mPa \text{ s}^\S, C = 0.7$ $\rho = 852 \text{ kg}/m^3$	$b = 0.0106$ $D = 0.966$
(DII) Tears of wine: new insights on an old phenomenon (cognac) (Venerus and Simavilla) [VS15]	$\gamma = 0.032N/m^{**}$ $\tau = 0.06Pa, \alpha = 45^\circ$ $\mu = 2.35mPa \text{ s}^\S, C = 0.35$ $\rho = 926 \text{ kg}/m^3$	$D = 0.0346$

** refers to surface tension interpolated from [VAN95] § refers to viscosity interpolated from [Vis]

† refers to precursor height taken from [FC92]

REFERENCES

- [ACB17] M. Arjovsky, S. Chintala, and L. Bottou. “Wasserstein GAN.” *arXiv:1701.07875 [cs, stat]*, 2017.
- [ACB19] D. Arpit, V. Campos, and Y. Bengio. “How to initialize your network? robust initialization for weightnorm & resnets.” In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pp. 10902–10911. Curran Associates, Inc., 2019.
- [ADG16] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas. “Learning to learn by gradient descent by gradient descent.” *arXiv preprint arXiv:1606.04474*, 2016.
- [ADH19] S. Arora, S. Du, W. Hu, Z. Li, and R. Wang. “Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks.” In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 322–332. PMLR, 2019.
- [AGR20] A. Achille, A. Golatkar, A. Ravichandran, M. Polito, and S. Soatto. “Lqf: Linear quadratic fine-tuning.” *arXiv preprint arXiv:2012.11140*, 2020.
- [AGS05] L. Ambrosio, N. Gigli, and Savaré Giuseppe. *Gradient Flows: In Metric Spaces and in the Space of Probability Measures*. Birkhäuser Basel, Basel, 2005.
- [AL18] J. Adler and S. Lunz. “Banach Wasserstein GAN.” *arXiv:1806.06621 [cs, math]*, 2018.
- [ALL19a] Z. Allen-Zhu, Y. Li, and Y. Liang. “Learning and generalization in overparameterized neural networks, going beyond two layers.” In *Advances in Neural Information Processing Systems 32*, pp. 6158–6169. 2019a.
- [ALL19b] S. Arora, Z. Li, and K. Lyu. “Theoretical analysis of auto rate-tuning by batch normalization.” In *International Conference on Learning Representations*, 2019b.
- [ALS19] Z. Allen-Zhu, Y. Li, and Z. Song. “A convergence theory for deep learning via over-parameterization.” In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 242–252. PMLR, 2019.
- [AMS97] C. G. Atkeson, A. W. Moore, and S. Schaal. “Locally weighted learning.” *Lazy learning*, pp. 11–73, 1997.
- [Ano22] Anonymous. “DIVA: Dataset derivative of a learning task.” In *Submitted to The Tenth International Conference on Learning Representations*, 2022. under review.

- [App15] Applied Science. “The science of wineglass tears (or wine legs).”, 2015. You Tube Video <https://www.youtube.com/watch?v=s6w0tSg-msk>.
- [ASY19] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. “Optuna: A next-generation hyperparameter optimization framework.” In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.
- [BB97] A. L. Bertozzi and M. P. Brenner. “Linear stability and transient growth in driven contact lines.” *Physics of Fluids*, **9**(3):530–539, 1997.
- [BBB99] M. Birattari, G. Bontempi, and H. Bersini. “Lazy learning meets the recursive least squares algorithm.” *Advances in neural information processing systems*, pp. 375–381, 1999.
- [BBG20] H. S. Behl, A. G. Baydin, R. Gal, P. H. Torr, and V. Vineet. “Autosimulate:(quickly) learning synthetic data generation.” In *European Conference on Computer Vision*, pp. 255–271. Springer, 2020.
- [BBL17] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. “Geometric deep learning: Going beyond euclidean data.” *IEEE Signal Processing Magazine*, **34**(4):18–42, 2017.
- [BBN19] D. Brickley, M. Burgess, and N. Noy. “Google dataset search: Building a search engine for datasets in an open web ecosystem.” In *The World Wide Web Conference*, pp. 1365–1375, 2019.
- [BF99] C. E. Brodley and M. A. Friedl. “Identifying mislabeled training data.” *Journal of artificial intelligence research*, **11**:131–167, 1999.
- [BF12] A. L. Bertozzi and A. Flenner. “Diffuse interface models on graphs for classification of high dimensional data.” *Multiscale Modeling & Simulation*, **10**(3):1090–1118, 2012.
- [BFL17] D. Balduzzi, M. Frean, L. Leary, J. Lewis, K. W.-D. Ma, and B. McWilliams. “The shattered gradients problem: If resnets are the answer, then what is the question?” In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 342–350. JMLR. org, 2017.
- [BGS18] N. Bjorck, C. P. Gomes, B. Selman, and K. Q. Weinberger. “Understanding batch normalization.” In *Advances in Neural Information Processing Systems 31*, pp. 7694–7705. 2018.
- [Bis95] C. M. Bishop. “Training with noise is equivalent to Tikhonov regularization.” *Neural Computation*, **7**(1):108–116, 1995.

- [BJG17] E. Bernton, P. E. Jacob, M. Gerber, and C. P. Robert. “Inference in generative models using the wasserstein distance.” *arXiv preprint arXiv:1701.05146*, 2017.
- [BKH16] J. L. Ba, J. R. Kiros, and G. E. Hinton. “Layer normalization.” *Deep Learning Symposium, NIPS-2016*, 2016.
- [BL19] J. Byrd and Z. Lipton. “What is the effect of importance weighting in deep learning?” In *International Conference on Machine Learning*, pp. 872–881. PMLR, 2019.
- [BMF98] A. L. Bertozzi, A. Münch, X. Fanton, and A. M. Cazabat. “Contact line stability and “undercompressive shocks” in driven thin film flow.” *Physical Review Letters*, **81**(23):5169, 1998.
- [BMS99] A. L. Bertozzi, A. Münch, and M. Shearer. “Undercompressive shocks in thin film flows.” *Physica D: Nonlinear Phenomena*, **134**(4):431–464, 1999.
- [BSB05] M. Bowen, J. Sur, A. L. Bertozzi, and R. P. Behringer. “Nonlinear dynamics of two-dimensional undercompressive shocks.” *Physica D: Nonlinear Phenomena*, **209**(1-4):36–48, 2005.
- [Caw06] G. C. Cawley. “Leave-one-out cross-validation based model selection criteria for weighted ls-svms.” In *The 2006 IEEE international joint conference on neural network proceedings*, pp. 1661–1668. IEEE, 2006.
- [CBG17] M. Cissé, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier. “Parseval networks: Improving robustness to adversarial examples.” In *ICML 34*, pp. 854–863, 2017.
- [CC93] P. Carles and A.-M. Cazabat. “The thickness of surface-tension-gradient-driven spreading films.” *Journal of colloid and interface science*, **157**(1):196–201, 1993.
- [CCC20] C.-Y. Chen, C.-H. Chang, and E. Y. Chang. “Hypernetwork-based augmentation.” *arXiv preprint arXiv:2006.06320*, 2020.
- [CG19] Y. Cao and Q. Gu. “Generalization bounds of stochastic gradient descent for wide and deep neural networks.” In *Advances in Neural Information Processing Systems 32*, pp. 10836–10846. 2019.
- [CG20] Y. Cao and Q. Gu. “Generalization error bounds of gradient descent for learning over-parameterized deep ReLU networks.” In *AAAI*, 2020.
- [CHL12] S.-N. Chow, W. Huang, Y. Li, and H. Zhou. “Fokker–Planck Equations for a Free Energy Functional or Markov Process on a Graph.” *Archive for Rational Mechanics and Analysis*, **203**(3):969–1008, 2012.

- [CJL19] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie. “Class-balanced loss based on effective number of samples.” In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9268–9277, 2019.
- [CKB17] N. Carlini, G. Katz, C. Barrett, and D. L. Dill. “Provably minimally-distorted adversarial examples.” *CoRR*, **abs/1709.10207**, 2017.
- [CLM17] H.-S. Chang, E. Learned-Miller, and A. McCallum. “Active bias: Training more accurate neural networks by emphasizing high variance samples.” *Advances in Neural Information Processing Systems*, **30**:1002–1012, 2017.
- [CLS19] Y. Cai, Q. Li, and Z. Shen. “A quantitative analysis of the effect of batch normalization on gradient descent.” In *International Conference on Machine Learning*, pp. 882–890, 2019.
- [CLZ18] S.-N. Chow, W. Li, and H. Zhou. “Entropy dissipation of Fokker-Planck equations on graphs.” *Discrete & Continuous Dynamical Systems, series A*, 2018.
- [COB19] L. Chizat, E. Oyallon, and F. Bach. “On lazy training in differentiable programming.” In *Advances in Neural Information Processing Systems 32*, pp. 2937–2947. 2019.
- [COM15] COMSOL. “Marangoni effect: Tears of wine (and rum).”, 2015. You Tube Video <https://www.youtube.com/watch?v=i2rqCRMN4LQ>.
- [CT10] G. C. Cawley and N. L. Talbot. “On over-fitting in model selection and subsequent selection bias in performance evaluation.” *The Journal of Machine Learning Research*, **11**:2079–2107, 2010.
- [CYM19] C. Coleman, C. Yeh, S. Mussmann, B. Mirzasoleiman, P. Bailis, P. Liang, J. Leskovec, and M. Zaharia. “Selection via proxy: Efficient data selection for deep learning.” In *International Conference on Learning Representations*, 2019.
- [CZM18] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. “Autoaugment: Learning augmentation policies from data.” *arXiv preprint arXiv:1805.09501*, 2018.
- [Dan15] Dan Quinn. “Why does wine cry?”, 2015. You Tube Video <https://www.youtube.com/watch?v=tgrTbvSnE50>.
- [DAR21] A. Deshpande, A. Achille, A. Ravichandran, H. Li, L. Zancato, C. Fowlkes, R. Bhotika, S. Soatto, and P. Perona. “A linearized framework and a new benchmark for model selection for fine-tuning.” *arXiv preprint arXiv:2102.00084*, 2021.

- [DDS09] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. “Imagenet: A large-scale hierarchical image database.” In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.
- [DGM20] Y. Dukler, Q. Gu, and G. Montufar. “Optimization theory for ReLU neural networks trained with normalization layers.” In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2751–2760. PMLR, 2020.
- [DGZ17] Q. Dong, S. Gong, and X. Zhu. “Class rectification hard mining for imbalanced deep learning.” In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1851–1860, 2017.
- [DJF20] Y. Dukler, H. Ji, C. Falcon, and A. L. Bertozzi. “Theory for undercompressive shocks in tears of wine.” *Physical Review Fluids*, 5(3):034002, 2020.
- [DLL19a] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai. “Gradient descent finds global minima of deep neural networks.” In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1675–1685, Long Beach, California, USA, 2019a. PMLR.
- [DLL19b] Y. Dukler, W. Li, A. Lin, and G. Montufar. “Wasserstein of Wasserstein loss for learning generative models.” In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1716–1725, Long Beach, California, USA, 2019b. PMLR.
- [DLT18] S. S. Du, J. D. Lee, and Y. Tian. “When is a convolutional filter easy to learn?” In *International Conference on Learning Representations*, 2018.
- [DMM19] P. Dognin, I. Melnyk, Y. Mroueh, J. Ross, C. D. Santos, and T. Sercu. “Wasserstein barycenter model ensembling.” *arXiv preprint arXiv:1902.04999*, 2019.
- [DZP19] S. S. Du, X. Zhai, B. Póczos, and A. Singh. “Gradient descent provably optimizes over-parameterized neural networks.” In *International Conference on Learning Representations*, 2019.
- [DZS18] I. Deshpande, Z. Zhang, and A. G. Schwing. “Generative modeling using the sliced wasserstein distance.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3483–3491, 2018.
- [EMH19] T. Elsken, J. H. Metzen, F. Hutter, et al. “Neural architecture search: A survey.” *J. Mach. Learn. Res.*, 20(55):1–21, 2019.
- [EY18] B. Engquist and Y. Yang. “Seismic imaging and optimal transport.” *arXiv:1808.04801*, 2018.

- [FC92] J. Fournier and A. Cazabat. “Tears of wine.” *EPL (Europhysics Letters)*, **20**(6):517, 1992.
- [FC98] X. Fanton and A. Cazabat. “Spreading and instabilities induced by a solutal marangoni effect.” *Langmuir*, **14**(9):2554–2561, 1998.
- [FCQ96] X. Fanton, A. Cazabat, and D. Quéré. “Thickness and shape of films driven by a marangoni flow.” *Langmuir*, **12**(24):5875–5880, 1996.
- [FKE19] M. Feurer, A. Klein, K. Eggenberger, J. T. Springenberg, M. Blum, and F. Hutter. “Auto-sklearn: efficient and robust automated machine learning.” In *Automated Machine Learning*, pp. 113–134. Springer, Cham, 2019.
- [FOA19] C. Finlay, A. M. Oberman, and B. Abbasi. “Improved robustness to adversarial examples using lipschitz regularization of the loss.”, 2019.
- [FZM15] C. Frogner, C. Zhang, H. Mobahi, M. Araya-Polo, and T. Poggio. “Learning with a Wasserstein Loss.” *arXiv:1506.05439 [cs, stat]*, 2015.
- [GAA17] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. “Improved training of wasserstein GANs.” In *Advances in Neural Information Processing Systems 30*, pp. 5767–5777. Curran Associates, Inc., 2017.
- [GBS06] J. B. Greer, A. L. Bertozzi, and G. Sapiro. “Fourth order partial differential equations on general geometries.” *Journal of Computational Physics*, **216**(1):216–246, 2006.
- [GCK17] R. Gao, X. Chen, and A. J. Kleywegt. “Wasserstein distributional robustness and regularization in statistical learning.” *arXiv preprint arXiv:1712.06050*, 2017.
- [GG17] I. Gitman and B. Ginsburg. “Comparison of batch normalization and weight normalization algorithms for the large-scale image classification.” *arXiv preprint arXiv:1709.08145*, 2017.
- [GHL15] J. Gu, B. Hua, and S. Liu. “Spectral distances on graphs.” *Discrete Applied Mathematics*, **190**:56–74, 2015.
- [GHP07] G. Griffin, A. Holub, and P. Perona. “Caltech-256 object category dataset.” 2007.
- [GLO19] W. Gangbo, W. Li, S. Osher, and M. Puthawala. “Unnormalized optimal transport.” *Journal of Computational Physics*, **399**:108940, 2019.
- [GPC18] A. Genevay, G. Peyre, and M. Cuturi. “Learning generative models with sinkhorn divergences.” In *International Conference on Artificial Intelligence and Statistics*, pp. 1608–1617, 2018.

- [GRC18] C. Guo, M. Rana, M. Cisse, and L. van der Maaten. “Countering adversarial images using input transformations.” In *International Conference on Learning Representations*, 2018.
- [GRP01] A. Golovin, B. Rubinstein, and L. Pismen. “Effect of van der waals interactions on the fingering instability of thermally driven thin wetting films.” *Langmuir*, **17**(13):3930–3936, 2001.
- [GS93] P. J. Green and B. W. Silverman. *Nonparametric regression and generalized linear models: a roughness penalty approach*. Crc Press, 1993.
- [GSS15] I. J. Goodfellow, J. Shlens, and C. Szegedy. “Explaining and harnessing adversarial examples.” In *ICLR*, 2015.
- [HA17] M. Hein and M. Andriushchenko. “Formal guarantees on the robustness of a classifier against adversarial manipulation.” In *Advances in Neural Information Processing Systems 30*, pp. 2266–2276. Curran Associates, Inc., 2017.
- [HB01] A. Hosoi and J. W. Bush. “Evaporative instabilities in climbing films.” *Journal of Fluid Mechanics*, **442**:217–239, 2001.
- [HBG18] E. Hoffer, R. Banner, I. Golan, and D. Soudry. “Norm matters: efficient and accurate normalization schemes in deep networks.” In *Advances in Neural Information Processing Systems 31*, pp. 2160–2170. 2018.
- [HCH07] X. Hong, S. Chen, and C. J. Harris. “A kernel-based two-class classifier for imbalanced data sets.” *IEEE Transactions on neural networks*, **18**(1):28–41, 2007.
- [HHS17] E. Hoffer, I. Hubara, and D. Soudry. “Train longer, generalize better: closing the generalization gap in large batch training of neural networks.” In *Advances in Neural Information Processing Systems 30*, pp. 1731–1741. 2017.
- [HJS20] M. Hwang, Y. Jeong, and W. Sung. “Data distribution search to select core-set for machine learning.” In *Proceedings of the 9th International Conference on Smart Media & Applications (SMA 2020), Jeju, Korea*, pp. 17–19, 2020.
- [HKF18] K. M. Hosny, M. A. Kassem, and M. M. Foad. “Skin cancer classification using deep learning and transfer learning.” In *2018 9th Cairo International Biomedical Engineering Conference (CIBEC)*, pp. 90–93. IEEE, 2018.
- [HLL19] C. Huang, Y. Li, C. C. Loy, and X. Tang. “Deep imbalanced learning for face recognition and attribute prediction.” *IEEE transactions on pattern analysis and machine intelligence*, **42**(11):2781–2794, 2019.
- [HS17] C. Hwang and J. Shim. “Geographically weighted least squares-support vector machine.” *Journal of the Korean Data and Information Science Society*, **28**(1):227–235, 2017.

- [HZC21] X. He, K. Zhao, and X. Chu. “Automl: A survey of the state-of-the-art.” *Knowledge-Based Systems*, **212**:106622, 2021.
- [IS15] S. Ioffe and C. Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift.” In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 448–456. PMLR, 2015.
- [JF18] S. Jenni and P. Favaro. “Deep bilevel learning.” In *Proceedings of the European conference on computer vision (ECCV)*, pp. 618–633, 2018.
- [JG18] D. Jakubovitz and R. Giryes. “Improving DNN robustness to adversarial attacks using jacobian regularization.” *CoRR*, **abs/1803.08680**, 2018.
- [JGH18] A. Jacot, F. Gabriel, and C. Hongler. “Neural tangent kernel: Convergence and generalization in neural networks.” In *Advances in Neural Information Processing Systems 31*, pp. 8571–8580. 2018.
- [JHS20] Y. Jeong, M. Hwang, and W. Sung. “Dataset distillation for core training set construction.” 2020.
- [JVE20] M. Joneidi, S. Vahidian, A. Esmaeili, W. Wang, N. Rahnavard, B. Lin, and M. Shah. “Select to better learn: Fast and accurate deep learning using data selection from nonlinear manifolds.” In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7819–7829, 2020.
- [KDL19] J. Kohler, H. Daneshmand, A. Lucchi, T. Hofmann, M. Zhou, and K. Neymeyr. “Exponential convergence rates for batch normalization: The power of length-direction decoupling in non-convex optimization.” In *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pp. 806–815. PMLR, 2019.
- [KGB17] A. Kurakin, I. J. Goodfellow, and S. Bengio. “Adversarial machine learning at scale.” In *ICLR*. OpenReview.net, 2017.
- [KL17] P. W. Koh and P. Liang. “Understanding black-box predictions via influence functions.” In *International Conference on Machine Learning*, pp. 1885–1894. PMLR, 2017.
- [KSD13] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. “3d object representations for fine-grained categorization.” In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.
- [KSM21] K. Killamsetty, D. Sivasubramanian, B. Mirzasoleiman, G. Ramakrishnan, A. De, and R. K. Iyer. “GRAD-MATCH: A gradient matching based data subset selection for efficient learning.” *CoRR*, **abs/2103.00123**, 2021.

- [Laf88] J. D. Lafferty. “The Density Manifold and Configuration Space Quantization.” *Transactions of the American Mathematical Society*, **305**(2):699–741, 1988.
- [LCS19] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei. “Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation.” In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 82–92, 2019.
- [LCY20] H. Li, P. Chaudhari, H. Yang, M. Lam, A. Ravichandran, R. Bhotika, and S. Soatto. “Rethinking the hyperparameters for fine-tuning.” *arXiv preprint arXiv:2002.11770*, 2020.
- [LDL19] A. T. Lin, Y. Dukler, W. Li, and G. Montúfar. “Wasserstein diffusion tikhonov regularization.” *arXiv preprint arXiv:1909.06860*, 2019.
- [LGG17] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. “Focal loss for dense object detection.” In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.
- [LHH21] A. Liu, Z. Huang, Z. Huang, and N. Wang. “Direct differentiable augmentation search.” *arXiv preprint arXiv:2104.04282*, 2021.
- [LHW20] Y. Li, G. Hu, Y. Wang, T. M. Hospedales, N. M. Robertson, and Y. Yang. “DADA: differentiable automatic data augmentation.” 2020.
- [Li18] W. Li. “Geometry of probability simplex via optimal transport.” *arXiv:1803.06360 [math]*, 2018.
- [Liu08] J. S. Liu. *Monte Carlo strategies in scientific computing*. Springer Science & Business Media, 2008.
- [LJD17] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. “Hyperband: A novel bandit-based approach to hyperparameter optimization.” *The Journal of Machine Learning Research*, **18**(1):6765–6816, 2017.
- [LKK19] S. Lim, I. Kim, T. Kim, C. Kim, and S. Kim. “Fast autoaugment.” *arXiv preprint arXiv:1905.00397*, 2019.
- [LL18] Y. Li and Y. Liang. “Learning overparameterized neural networks via stochastic gradient descent on structured data.” In *Advances in Neural Information Processing Systems 31*, pp. 8157–8166. 2018.
- [LLD18] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu. “Defense against adversarial attacks using high-level representation guided denoiser.” In *CVPR*, pp. 1778–1787, 2018.
- [LLO18] A. Lin, W. Li, S. Osher, and G. Montúfar. “Wasserstein proximal of GANs.” *CAM report 18-53*, 2018.

- [LLW19] B. Li, Y. Liu, and X. Wang. “Gradient harmonized single-stage detector.” In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 8577–8584, 2019.
- [LM18a] W. Li and G. Montúfar. “Natural gradient via optimal transport.” *Information Geometry*, **1**(2):181–214, 2018a.
- [LM18b] W. Li and G. Montúfar. “Ricci curvature for parametric statistics via optimal transport.” *arXiv:1807.07095*, 2018b.
- [LSY18] H. Liu, K. Simonyan, and Y. Yang. “Darts: Differentiable architecture search.” *arXiv preprint arXiv:1806.09055*, 2018.
- [LWS18] P. Luo, X. Wang, W. Shao, and Z. Peng. “Understanding regularization in batch normalization.” *arXiv preprint arXiv:1809.00846*, 2018.
- [LY17] Y. Li and Y. Yuan. “Convergence analysis of two-layer neural networks with ReLU activation.” In *Advances in Neural Information Processing Systems 30*, pp. 597–607. 2017.
- [Maa11] J. Maas. “Gradient flows of the entropy for finite Markov chains.” *Journal of Functional Analysis*, **261**(8):2250–2292, 2011.
- [MB99] A. Münch and A. Bertozzi. “Rarefaction–undercompressive fronts in driven films.” *Physics of Fluids*, **11**(10):2812–2814, 1999.
- [MCY18] R. Meng, Q. Cui, and C. Yuan. “A survey of image information hiding algorithms based on deep learning.” *Computer Modeling in Engineering & Sciences*, **117**:425–454, 2018.
- [ME06] A. Münch and P. Evans. “Interaction of advancing fronts and meniscus profiles formed by surface-tension-gradient-driven liquid films.” *SIAM Journal on Applied Mathematics*, **66**(5):1610–1631, 2006.
- [MFF16] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. “Deepfool: A simple and accurate method to fool deep neural networks.” In *CVPR*, pp. 2574–2582. IEEE Computer Society, 2016.
- [MFF17] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. “Universal adversarial perturbations.” In *CVPR*, pp. 86–94. IEEE Computer Society, 2017.
- [MG15] J. Martens and R. Grosse. “Optimizing neural networks with Kronecker-factored approximate curvature.” In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2408–2417. PMLR, 2015.

- [MGM18] R. Mormont, P. Geurts, and R. Marée. “Comparison of deep transfer learning strategies for digital pathology.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 2262–2271, 2018.
- [MHJ92] A. W. Moore, D. J. Hill, and M. P. Johnson. “An empirical investigation of brute force to choose features, smoothers and function approximators.” In *Computational Learning Theory and Natural Learning Systems*. MIT Press, 1992.
- [Mie11] A. Mielke. “A Gradient Structure for Reaction–diffusion Systems and for Energy-Drift-Diffusion Systems.” *Nonlinearity*, **24**(4):1329, 2011.
- [MKR13] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. “Fine-grained visual classification of aircraft.” Technical report, 2013.
- [MLL20] F. Mu, Y. Liang, and Y. Li. “Gradients as features for deep representation learning.” *arXiv preprint arXiv:2004.05529*, 2020.
- [MMC16] G. Montavon, K.-R. Müller, and M. Cuturi. “Wasserstein Training of Restricted Boltzmann Machines.” In *Advances in Neural Information Processing Systems 29*, pp. 3718–3726. Curran Associates, Inc., 2016.
- [MSG17] Y. Mroueh, T. Sercu, and V. Goel. “McGan: Mean and covariance feature matching GAN.” In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2527–2535. PMLR, 2017.
- [MU49] N. Metropolis and S. Ulam. “The monte carlo method.” *Journal of the American statistical association*, **44**(247):335–341, 1949.
- [Mun00] A. Münch. “Shock transitions in marangoni gravity-driven thin-film flow.” *Nonlinearity*, **13**(3):731, 2000.
- [Mun03] A. Münch. “Pinch-off transition in marangoni-driven thin films.” *Physical Review Letters*, **91**(1):016105, 2003.
- [NH10] V. Nair and G. E. Hinton. “Rectified linear units improve restricted Boltzmann machines.” In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.
- [NJT06] E. Nowak, F. Jurie, and B. Triggs. “Sampling strategies for bag-of-features image classification.” In *European Conference on Computer Vision*, pp. 490–503. Springer, 2006.
- [NRS21] N. Nikolova, R. M. Rodríguez, M. Symes, D. Toneva, K. Kolev, and K. Tenekedjiev. “Outlier detection algorithms over fuzzy data with weighted least squares.” *International Journal of Fuzzy Systems*, pp. 1–23, 2021.

- [NWC11] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. “Reading digits in natural images with unsupervised feature learning.” 2011.
- [NWL18] A. Nikolov, D. Wasan, and J. Lee. “Tears of wine: The dance of the droplets.” *Advances in Colloid and Interface Science*, 2018.
- [NYC15] A. M. Nguyen, J. Yosinski, and J. Clune. “Deep neural networks are easily fooled: High confidence predictions for unrecognizable images.” In *CVPR*, pp. 427–436, 2015.
- [NZ08] M.-E. Nilsback and A. Zisserman. “Automated flower classification over a large number of classes.” In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pp. 722–729. IEEE, 2008.
- [ODB97] A. Oron, S. H. Davis, and S. G. Bankoff. “Long-scale evolution of thin liquid films.” *Reviews of modern physics*, **69**(3):931, 1997.
- [OS19] S. Oymak and M. Soltanolkotabi. “Towards moderate overparameterization: global convergence guarantees for training shallow neural networks.” *arXiv preprint arXiv:1902.04674*, 2019.
- [Ott01] F. Otto. “The Geometry of Dissipative Evolution Equations: The Porous Medium Equation.” *Communications in Partial Differential Equations*, **26**(1-2):101–174, 2001.
- [PC19] G. Peyré, M. Cuturi, et al. “Computational optimal transport.” *Foundations and Trends® in Machine Learning*, **11**(5-6):355–607, 2019.
- [PFL17] H. Petzka, A. Fischer, and D. Lukovnicov. “On the regularization of Wasserstein GANs.” *arXiv:1709.08894 [cs, stat]*, 2017.
- [PHO18] M. A. Puthawala, C. D. Hauck, and S. J. Osher. “Diagnosing forward operator error using optimal transport.” *arXiv:1810.12993*, 2018.
- [PLS20] G. Pruthi, F. Liu, M. Sundararajan, and S. Kale. “Estimating training data influence by tracking gradient descent.” *arXiv preprint arXiv:2002.08484*, 2020.
- [PVG11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. “Scikit-learn: Machine learning in python.” *the Journal of machine Learning research*, **12**:2825–2830, 2011.
- [PVZ12] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. Jawahar. “Cats and dogs.” In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3498–3505. IEEE, 2012.

- [QLL10] T. Quan, X. Liu, and Q. Liu. “Weighted least squares support vector machine local region method for nonlinear time series prediction.” *Appl. Soft Comput.*, **10**(2):562–566, 2010.
- [QT09] A. Quattoni and A. Torralba. “Recognizing indoor scenes.” In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 413–420. IEEE, 2009.
- [RKH21] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. “Learning transferable visual models from natural language supervision.” *arXiv preprint arXiv:2103.00020*, 2021.
- [RL07] R. M. Rifkin and R. A. Lippert. “Notes on regularized least squares.” 2007.
- [RRC17] E. Rezende, G. Ruppert, T. Carvalho, F. Ramos, and P. De Geus. “Malicious software classification using transfer learning of resnet-50 deep neural network.” In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1011–1014. IEEE, 2017.
- [RRS02] R. V. Roy, A. J. Roberts, and M. E. Simpson. “A lubrication model of coating flows over a curved substrate in space.” *Journal of Fluid Mechanics*, **454**:235–261, 2002.
- [RS09] M.-K. von Renesse and K.-T. Sturm. “Entropic measure and Wasserstein diffusion.” *Ann. Probab.*, **37**(3):1114–1191, 2009.
- [RTG00] Y. Rubner, C. Tomasi, and L. J. Guibas. “The Earth Mover’s Distance as a Metric for Image Retrieval.” *International Journal of Computer Vision*, **40**(2):99–121, 2000.
- [RZY18] M. Ren, W. Zeng, B. Yang, and R. Urtasun. “Learning to reweight examples for robust deep learning.” In *International Conference on Machine Learning*, pp. 4334–4343. PMLR, 2018.
- [SBB03] J. Sur, A. L. Bertozzi, and R. P. Behringer. “Reverse undercompressive shock structures in driven thin film flow.” *Physical Review Letters*, **90**(12):126105, 2003.
- [SC00] M. Schneemilch and A. Cazabat. “Shock separation in wetting films driven by thermal gradients.” *Langmuir*, **16**(25):9850–9856, 2000.
- [SDP15] J. Solomon, F. De Goes, G. Peyré, M. Cuturi, A. Butscher, A. Nguyen, T. Du, and L. Guibas. “Convolutional Wasserstein distances: Efficient optimal transportation on geometric domains.” *ACM Transactions on Graphics (TOG)*, **34**(4):66, 2015.
- [SGG16] A. Shrivastava, A. Gupta, and R. Girshick. “Training region-based object detectors with online hard example mining.” In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 761–769, 2016.

- [SHD18] S. P. Singh, A. Hug, A. Dieuleveut, and M. Jaggi. “Wasserstein is all you need.” *arXiv preprint arXiv:1808.09663*, 2018.
- [SHS19] A. Shafahi, W. R. Huang, C. Studer, S. Feizi, and T. Goldstein. “Are adversarial examples inevitable?” In *ICLR*, 2019.
- [SK16] T. Salimans and D. P. Kingma. “Weight normalization: A simple reparameterization to accelerate training of deep neural networks.” In *Advances in Neural Information Processing Systems 29*, pp. 901–909. 2016.
- [SKC18] P. Samangouei, M. Kabkab, and R. Chellappa. “Defense-GAN: Protecting classifiers against adversarial attacks using generative models.” In *ICLR*, 2018.
- [SKE17] S. Shafieezadeh-Abadeh, D. Kuhn, and P. M. Esfahani. “Regularization via mass transportation.” *arXiv preprint arXiv:1710.10016*, 2017.
- [SLF16] A. Schlichting, V. Laschos, M. Fathi, and M. Erbar. “Gradient flow structure for McKean-Vlasov equations on discrete spaces.” *Discrete and Continuous Dynamical Systems*, **36**(12):6799–6833, 2016.
- [SRG14] J. Solomon, R. Rustamov, L. Guibas, and A. Butscher. “Earth mover’s distances on discrete surfaces.” *ACM Transactions on Graphics (TOG)*, **33**(4):67, 2014.
- [STI18] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry. “How does batch normalization help optimization?” In *Advances in Neural Information Processing Systems 31*, pp. 2483–2493. 2018.
- [Sto77] M. Stone. “Asymptotics for and against cross-validation.” *Biometrika*, pp. 29–35, 1977.
- [SYZ19] H. Salman, G. Yang, H. Zhang, C.-J. Hsieh, and P. Zhang. “A convex relaxation barrier to tight robustness verification of neural networks.” *ArXiv*, **abs/1902.08722**, 2019.
- [SZS14] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. “Intriguing properties of neural networks.” In *ICLR*, 2014.
- [TB18] R. Trichet and F. Bremond. “Dataset optimization for real-time pedestrian detection.” *IEEE access*, **6**:7719–7727, 2018.
- [Tho55] J. Thomson. “Xlii. on certain curious motions observable at the surfaces of wine and other alcoholic liquors.” *Philosophical Magazine Series 4*, **10**(67):330–333, 1855.
- [Tia17] Y. Tian. “An analytical formula of population gradient for two-layered ReLU network and its applications in convergence and critical point analysis.” In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3404–3413. JMLR. org, 2017.

- [TL19] M. Tan and Q. Le. “Efficientnet: Rethinking model scaling for convolutional neural networks.” In *International Conference on Machine Learning*, pp. 6105–6114. PMLR, 2019.
- [TMW20] M. Thapa, S. B. Mulani, and R. W. Walters. “Adaptive weighted least-squares polynomial chaos expansion with basis adaptivity and sequential adaptive sampling.” *Computer Methods in Applied Mechanics and Engineering*, **360**:112759, 2020.
- [VAN95] G. Vazquez, E. Alvarez, and J. M. Navaza. “Surface tension of alcohol water+ water from 20 to 50. degree. c.” *Journal of Chemical and Engineering Data*, **40**(3):611–614, 1995.
- [VEN95] R. Vuilleumier, V. Ego, L. Neltner, and A. Cazabat. “Tears of wine: the stationary state.” *Langmuir*, **11**(10):4117–4121, 1995.
- [Ver18] R. Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge University Press, 2018.
- [Vil09] C. Villani. *Optimal Transport: Old and New*. Number 338 in Grundlehren der mathematischen Wissenschaften. Springer, Berlin, 2009.
- [Vis] “Viscosity of two component mixtures.” <http://www.rheosense.com/applications/viscosity/two-component-mixtures>. Accessed: 2018-03-10.
- [VS15] D. C. Venerus and D. N. Simavilla. “Tears of wine: New insights on an old phenomenon.” *Scientific Reports*, **5**:16162, 2015.
- [VSP17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. “Attention is all you need.” In *Advances in neural information processing systems*, pp. 5998–6008, 2017.
- [WBM10] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. “Caltech-UCSD Birds 200.” Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.
- [WDW19] X. Wu, S. S. Du, and R. Ward. “Global convergence of adaptive gradient methods for an over-parameterized neural network.” *arXiv preprint arXiv:1902.07111*, 2019.
- [WDZ19] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer. “Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search.” In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10734–10742, 2019.
- [WHY08] W. Wen, Z. Hao, and X. Yang. “A heuristic weight-setting strategy and iteratively updating algorithm for weighted least-squares support vector regression.” *Neurocomputing*, **71**(16-18):3096–3103, 2008.

- [WLS18] B. Wang, A. Lin, Z. Shi, W. Zhu, P. Yin, A. L. Bertozzi, and S. J. Osher. “Adversarial defense via data dependent activation function and total variation minimization.” *CoRR*, **abs/1809.08516**, 2018.
- [WSK19] E. Wong, F. Schmidt, and Z. Kolter. “Wasserstein adversarial examples via projected Sinkhorn iterations.” In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6808–6817, Long Beach, California, USA, 2019. PMLR.
- [WWB18] X. Wu, R. Ward, and L. Bottou. “WNGrad: Learn the learning rate in gradient descent.” *arXiv preprint arXiv:1803.02865*, 2018.
- [WYS18] B. Wang, B. Yuan, Z. Shi, and S. J. Osher. “EnResNet: ResNet ensemble via the Feynman-Kac formalism.” *CoRR*, **abs/1811.10745**, 2018.
- [WZT18] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros. “Dataset distillation.” *arXiv preprint arXiv:1811.10959*, 2018.
- [XHZ19] J. Xue, J. Han, T. Zheng, J. Guo, and B. Wu. “Hard sample mining for the improved retraining of automatic speech recognition.” *arXiv preprint arXiv:1904.08031*, 2019.
- [XYR21] D. Xu, Y. Ye, and C. Ruan. “Understanding the role of importance weighting for deep learning.” In *International Conference on Learning Representations*, 2021.
- [YAF20] X. Yan, D. Acuna, and S. Fidler. “Neural data server: A large-scale search engine for transfer learning data.” In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3893–3902, 2020.
- [YGZ18] Z. Yan, Y. Guo, and C. Zhang. “Deep defense: Training dnns with improved adversarial robustness.” In *Advances in Neural Information Processing Systems 31*, pp. 419–428. Curran Associates, Inc., 2018.
- [YKO17] Y. Yoshida, R. Karakida, M. Okada, and S.-i. Amari. “Statistical mechanical analysis of online learning with weight normalization in single layer perceptron.” *Journal of the Physical Society of Japan*, **86**(4):044002, 2017.
- [YPR19] G. Yang, J. Pennington, V. Rao, J. Sohl-Dickstein, and S. S. Schoenholz. “A mean field theory of batch normalization.” In *International Conference on Learning Representations*, 2019.
- [YZW17] L. Yu, W. Zhang, J. Wang, and Y. Yu. “Seqgan: Sequence generative adversarial nets with policy gradient.” In *AAAI*, pp. 2852–2858, 2017.
- [ZBC11] M. Zheng, J. Bu, C. Chen, C. Wang, L. Zhang, G. Qiu, and D. Cai. “Graph regularized sparse coding for image representation.” *IEEE Transactions on Image Processing*, **20**(5):1327–1336, 2011.

- [ZCZ20] D. Zou, Y. Cao, D. Zhou, and Q. Gu. “Gradient descent optimizes over-parameterized deep ReLU networks.” *Machine Learning*, **109**(3):467–492, 2020.
- [ZG19] D. Zou and Q. Gu. “An improved analysis of training over-parameterized deep neural networks.” In *Advances in Neural Information Processing Systems 32*, pp. 2055–2064. 2019.
- [ZL16] B. Zoph and Q. V. Le. “Neural architecture search with reinforcement learning.” *arXiv preprint arXiv:1611.01578*, 2016.
- [ZLK17] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. “Places: A 10 million image database for scene recognition.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.
- [ZMG19] G. Zhang, J. Martens, and R. B. Grosse. “Fast convergence of natural gradient descent for over-parameterized neural networks.” In *Advances in Neural Information Processing Systems 32*, pp. 8082–8093. 2019.
- [ZML07] J. Zhang, M. Marszałek, S. Lazebnik, and C. Schmid. “Local features and kernels for classification of texture and object categories: A comprehensive study.” *International Journal of Computer Vision*, **73**(2):213–238, 2007.
- [ZNR17] V. Zantedeschi, M.-I. Nicolae, and A. Rawat. “Efficient defenses against adversarial attacks.” In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec ’17*, pp. 39–49, New York, NY, USA, 2017. ACM.
- [ZYJ19] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan. “Theoretically principled trade-off between robustness and accuracy.” *CoRR*, **abs/1901.08573**, 2019.