Fall 10-1-2021

Learning, Optimization and Data Translation with Deep Neural Networks

Ganlin Song
Yale University Graduate School of Arts and Sciences, glsong2021@gmail.com

Abstract

Learning, Optimization, and Data Translation with Deep Neural Networks

Ganlin Song

2021

Neural networks have been intensively studied as machine learning models and widely applied in various areas. This thesis investigates three problems related to the theory and application of neural networks. First, we analyze a learning scheme for neural networks that uses random weights in the backpropagation training algorithm, which is considered to be more biologically plausible than the standard training procedure. We establish theory that shows the convergence of the loss and the alignment between the forward weights of the network and the random weights used in the backward pass. Second, we study a family of optimization problems where the objective involves a trained generative network, with the goal of inverting the network. We introduce a novel algorithm that takes advantage of a sequential optimization technique to deal with the problem of non-convexity. The third part of this thesis is an application of modern neural network models to certain problems in neuroscience. We analyze data that contains two concurrent imaging modalities of the brain activity in mice, and build translation models to predict one modality the other. Our study is one of the first examples of advanced machine learning models applied to concurrent multi-model brain imaging data and demonstrates the potential of deep neural networks in the emerging area of neuroscience.

Learning, Optimization, and Data Translation with Deep Neural Networks

A Dissertation
Presented to the Faculty of the Graduate School
of
Yale University
in Candidacy for the Degree of
Doctor of Philosophy

By
Ganlin Song

Dissertation Director: John Lafferty

December 2021

Contents

List of Figures

List of Tables

Acknowledgements

I would first like to express my gratitude to my advisor, Professor John Lafferty. I want to thank him for his expert and tireless guidance over the past four years, and all of his creative ideas and enlightening feedback on my research projects. I start to realize how to become a good scholar after working with John, and I feel so lucky to be advised by him during my PhD studies.

I am also very grateful to Professor Zhou Fan for his valuable advice and efforts on surfing project. The discussions with him are always enjoyable, and his passionate attitude toward the research has inspired me a lot.

A special thank to Professor Evelyn Lake, for her great collaboration and constant support on biological knowledge and data processing.

I would like to thank Professor Harrison Zhou for his time spent on reviewing this dissertation.

I would also like to thank Ruitu Xu and Xinyi Zhong for their collaboration and friendship. Both of them are reliable and supportive partners to me.

Finally, I would like to acknowledge with gratitude, the support and love of my family. They have always been encouraging and believing in me. The completion of this dissertation would have never been possible without their support.

Chapter 1

Overview

Neural networks have been remarkably successful in machine learning, drawing on a wide range of algorithms that have been developed and studied for learning with neural networks over the past few decades. The learning problem is generically the problem of parameter estimation and typically is cast as an optimization problem. In fact, the vast majority of neural network approaches are based on optimizing a loss function, such as squared error loss or cross entropy loss. For example, backpropagation enables efficiently computing gradients of the parameters of the network, which allows stochastic gradient descent (SGD) to be used as one of the standard algorithms for neural network training. In this thesis, we first study a novel approach to the learning problem and show that learning is possible for certain neural networks without explicitly optimizing any loss function. This is motivated by the fact that the standard algorithms, which are based on backpropagation, are not biologically plausible. Specifically, backpropagation requires the use of the forward weights in the backward process, which is implausible in the brain given that the forward and backward weights of the neurons are not symmetric. We show that the use of random weights in backpropagation (Lillicrap et al., 2016) can actually carry out learning in over-parameterized two-layer networks, and show that with this learning procedure, the forward weights will be aligned with the random backward weights when proper regularization is applied. These results contribute to our understanding of the biological principles of information processing in the brain and shed light on various possibilities of training neural networks.

In the second part of the thesis, we focus on an optimization problem that arises after the neural network has been trained. In generative models that map a latent vector to a high dimensional output, it is important to be able to "invert" the model to find the latent vector that best explains the output. This type of inverse problem is often considered under a compressed sensing framework which aims to recover the unknown signal x from its (noisy) linear measurements $y = Ax + \epsilon$. If x is from a generative model G (Bora et al., 2017), one can approach this by minimizing an empirical risk function

$$\hat{z} = \operatorname*{argmin}_{z} \|AG(x) - y\|^2.$$

However, this optimization problem is generally non-convex due to the presence of the nonlinear mapping G, and the usual gradient descent algorithm can easily become stuck in a local optimum. To tackle this problem, we introduce an algorithm that uses a sequence of models G_t obtained during the course of training G. Gradient descent is sequentially applied on the empirical risk function for model G_t, where the converged point for model G_{t-1} is used as the initial point for model G_t. We call the algorithm "surfing" since it rides along the peak of the evolving surface of the (negative) empirical risk function. The algorithm is formalized and analyzed for a family of deep neural networks, and the experiments show that surfing can be used to find the global optimum even when direct gradient descent fails. In general, the surfing algorithm can be applied to any optimization problem that involves a trained neural network or other machine learning model.

Our third investigation is an application of modern deep neural networks to some emerging problems in neuroscience. Neuroscientists collect data by making measurements of brain activity using different technologies, including functional magnetic resonance imaging (fMRI), electroencephalogram (EEG) and magnetoencephalography (MEG). Methods based on tagging neurons with fluorescent markers enable researchers to observe neural activities at multiple scales. Related technologies for optogenetics allow researchers to modify the neural patterns and monitor the neuronal activities. These technologies provide scientific instruments that are used by neuroscientists to better understand the organizational principles that govern brain function. However, each of the data modalities has its own limitations and only reflects specific aspects of brain activity. Machine learning offers the potential to model across these modalities. In the third part of our thesis, we make one of the first investigations into advanced neural networks for this type of problem. We study the problem of mapping between calcium imaging, which shows neuronal activity on the cortical surface with fairly high resolution, and blood oxygen level dependent (BOLD) measurements using fMRI, which are low-resolution measurements of neural activity at multiple layers of the brain. We build translation models that exploit modern architectures for deep neural networks for images, and demonstrate the predictive power of these models from multiple perspectives. Our study is one of the first examples of this line of work, and highlights the challenges and potential of applying deep neural networks in future neuroscience research.

Chapter 2

Convergence and Alignment of Gradient Descent with Random Backpropagation Weights

Stochastic gradient descent with backpropagation is the workhorse of artificial neural networks. It has long been recognized that backpropagation fails to be a biologically plausible algorithm. Fundamentally, it is a non-local procedure—updating one neuron's synaptic weights requires knowledge of synaptic weights or receptive fields of downstream neurons. This limits the use of artificial neural networks as a tool for understanding the biological principles of information processing in the brain. Lillicrap et al. (2016) propose a more biologically plausible "feedback alignment" algorithm that uses random and fixed backpropagation weights, and show promising simulations. In this chapter we study the mathematical properties of the feedback alignment procedure by analyzing convergence and alignment for two-layer networks under squared error loss. In the overparameterized setting, we prove that the error converges to zero exponentially fast, and also that regularization is necessary in order for the parameters to become aligned with the random backpropagation weights.

Simulations are given that are consistent with this analysis and suggest further generalizations. These results contribute to our understanding of how biologically plausible algorithms might carry out weight learning in a manner different from Hebbian learning, with performance that is comparable with the full non-local backpropagation algorithm.

2.1 Introduction

The roots of artificial neural networks draw inspiration from networks of biological neurons (Rumelhart et al., 1986a; Elman et al., 1996; Medler, 1998). Grounded in simple abstractions of membrane potentials and firing, neural networks are increasingly being employed as a computational tool for better understanding the biological principles of information processing in the brain; examples include Yildirim et al. (2019) and Yamins and DiCarlo (2016). Even when full biological fidelity is not required, it can be useful to better align the computational abstraction with neuroscience principles.

Stochastic gradient descent has been a workhorse of artificial neural networks. Conveniently, calculation of gradients can be carried out using the backpropagation algorithm, where reverse mode automatic differentiation provides a powerful way of computing the derivatives for general architectures (Rumelhart et al., 1986b). Yet it has long been recognized that backpropagation fails to be a biologically plausible algorithm. Fundamentally, it is a non-local procedure—updating the weight between a presynaptic and postsynaptic neuron requires knowledge of the weights between the postsynaptic neuron and other neurons. No known biological mechanism exists for propagating information in this manner. This limits the use of artificial neural networks as a tool for understanding learning in the brain.

A wide range of approaches have been explored as a potential basis for learning and synaptic plasticity. Hebbian learning is the most fundamental procedure for adjusting weights, where repeated stimulation by a presynaptic neuron that results in the subsequent firing of the postsynapic neuron will result in an increased strength in the connection between the two cells (Hebb, 1961; Paulsen and Sejnowski, 2000). Several variants of Hebbian learning, some making connections to principal components analysis, have been proposed (Oja, 1982; Sejnowski and Tesauro, 1989; Sejnowski, 1999). In this chapter, our focus is on a formulation of Lillicrap et al. (2016) based on random backpropagation weights that are fixed during the learning process. Related proposals, including methods based on the use of differences of neuron activities, have been made in a series of recent papers (Akrout et al., 2019; Bellec et al., 2019; Lillicrap et al., 2020). A comparison of some of these methods is made by Bartunov et al. (2018).

The use of random feedback weights, which are not directly tied to the forward weights, removes issues of non-locality. However, it is not clear under what conditions optimization of error and learning can be successful. While Lillicrap et al. (2016) give suggestive simulations and some analysis for the linear case, it has been an open problem to explain the behavior of this algorithm for training the weights of a neural network. In this chapter we study the mathematical properties of the feedback alignment procedure by analyzing convergence and alignment for two-layer networks under squared error loss. In the overparameterized setting, we prove that the error converges to zero exponentially fast. We also show, unexpectedly, that the parameters become aligned with the random backpropagation weights only when regularization is used. Simulations are given that are consistent with this analysis and suggest further generalizations. The following section gives further background and an overview of our results.

2.2 Problem Statement and Overview of Results

In this section we provide a formulation of the backpropagation algorithm to establish notation and the context for our analysis. We then formulate the feedback alignment algorithm that uses random backpropagation weights. A high-level overview of our results is then presented, together with some of the intuition and proof techniques behind these results; we also contrast with what was known previously.

We mainly consider two-layer neural networks in the regression setting, specified by a family of functions $f : \mathbb{R}^d \to \mathbb{R}$ with input dimension d, sample size n, and p neurons in the hidden layer. For an input $x \in \mathbb{R}^d$, the network outputs

$$f(x) = \frac{1}{\sqrt{p}} \sum_{r=1}^{p} \beta_r \psi(w_r^\top x) = \frac{1}{\sqrt{p}} \beta^\top \psi(Wx), \tag{2.1}$$

where $W = (w_1, ..., w_p)^\top \in \mathbb{R}^{p\times d}$ and $\beta = (\beta_1, ..., \beta_p)^\top \in \mathbb{R}^p$ represent the feed-forward weights in the first and second layers, and ψ denotes an element-wise activation function. The scaling by $\sqrt{p}$ is simply for convenience in the analysis.

Given n input-response pairs $\{(x_i, y_i)\}_{i=1}^n$, the training objective is to minimize the squared error

$$\mathcal{L}(W, \beta) = \frac{1}{2} \sum_{i=1}^{n} \big(y_i - f(x_i)\big)^2. \tag{2.2}$$

Standard gradient descent attempts to minimize (2.2) by updating the feed-forward weights following gradient directions according to

$$\begin{aligned} \beta_r(t+1) &= \beta_r(t) - \eta \frac{\partial \mathcal{L}}{\partial \beta_r}(W(t), \beta(t)) \\ w_r(t+1) &= w_r(t) - \eta \frac{\partial \mathcal{L}}{\partial w_r}(W(t), \beta(t)), \end{aligned}$$

for each $r \in [p]$, where $\eta > 0$ denotes the step size. We initialize $\beta(0)$ and $w_r(0)$ as standard Gaussian vectors. We introduce the notation $f(t), e(t) \in \mathbb{R}^n$, with $f_i(t) =$

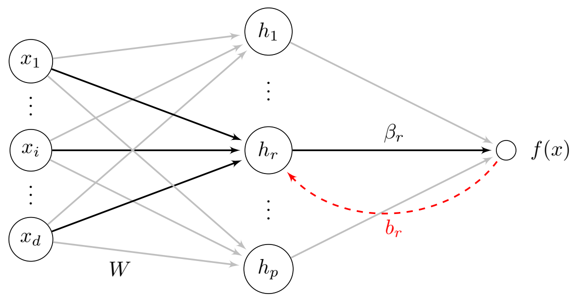


Figure 2.1: Standard backpropagation updates the first layer weights for a hidden node r with the second layer feedforward weight β_r. We study the procedure where the error is backpropagated instead using a fixed, random weight b_r.

$f(x_i)$ denoting the network output on input x_i when the weights are $W(t)$ and $\beta(t)$, and $e_i(t) = y_i - f_i(t)$ denoting the corresponding prediction error or residual. With this notation, the gradients are expressed as

$$\frac{\partial \mathcal{L}}{\partial \beta_r} = \frac{1}{\sqrt{p}} \sum_{i=1}^{n} e_i \psi(w_r^\top x_i), \qquad \frac{\partial \mathcal{L}}{\partial w_r} = \frac{1}{\sqrt{p}} \sum_{i=1}^{n} e_i \beta_r \psi'(w_r^\top x_i) x_i.$$

Here it is seen that the the gradient of the first-layer weights $\frac{\partial \mathcal{L}}{\partial w_r}$ involves not only the local input x_i and the change in the response of the r-th neuron, but also the backpropagated error signal $e_i\beta_r$. The appearance of β_r is, of course, due to the chain rule; but in effect it requires that the forward weights between layers are identical to the backward weights under error propagation. There is no evidence of biological mechanisms that would enable such "synaptic symmetry."

In the *feedback alignment* procedure of (Lillicrap et al., 2016), when updating the weights w_r, the error signal is weighted, and propagated backward, not by the second layer feedforward weights β, but rather by a random set of weights $b \in \mathbb{R}^p$ that are fixed during the course of training. Equivalently, the gradients for the first layer are replaced by the terms

$$\widetilde{\frac{\partial \mathcal{L}}{\partial w_r}} = \frac{1}{\sqrt{p}} \sum_{i=1}^{n} e_i b_r \psi'(w_r^\top x_i) x_i. \tag{2.3}$$

Note, however, that this update rule does not correspond to the gradient with respect to a modified loss function. The use of a random weight b_r when updating the first layer weights w_r does not violate locality, and could conceivably be implemented by biological mechanisms; we refer to Lillicrap et al. (2016); Bartunov et al. (2018); Lillicrap et al. (2020) for further discussion. A schematic of the relationship between the two algorithms is shown in Fig. 2.1.

Algorithm 1 Feedback Alignment on Two-layer Networks

```
Input: Dataset $\{(x_i, y_i)\}_{i=1}^n$, step size $\eta$
1: initialize $W$, $\beta$ and $b$ as Gaussian
2: while not converged do
3:     $\beta_r \leftarrow \beta_r - \frac{\eta}{\sqrt{p}} \sum_{i=1}^n e_i \psi(w_r^\top x_i)$
4:     $w_r \leftarrow w_r - \frac{\eta}{\sqrt{p}} \sum_{i=1}^n e_i b_r \psi'(w_r^\top x_i) x_i$
5:     for $r \in [p]$
6: end while
```

We can now summarize the main results that will be presented in this chapter. The first result shows that the error converges to zero when using random backpropagation weights.

- Under Gaussian initialization of the parameters, if the model is sufficiently over-parameterized with $p \gg n$, then the error converges to zero linearly. Moreover, the parameters satisfy $\|w_r(t) - w_r(0)\| = \widetilde{O}\big(\frac{n}{\sqrt{p}}\big)$ and $|\beta_r(t) - \beta_r(0)| = \widetilde{O}\big(\frac{n}{\sqrt{p}}\big)$.

The precise assumptions and statement of this result are given in Theorem 2.3.2. The proof shows in the over-parameterized regime that the weights only change by a small amount. While related to results for standard gradient descent, new methods are required because the "effective kernel" is not positive semi-definite.

We next turn to the issue of alignment of the second layer parameters β with the random backpropagation weights b. Such alignment was first observed in the original simulations of Lillicrap et al. (2016). With $h \in \mathbb{R}^p$ denoting the hidden layer of the two-layer network, the term $\delta_{\text{BP}}(h) \coloneqq \frac{\partial \mathcal{L}}{\partial h} = \frac{1}{\sqrt{p}}\beta \sum_{i=1}^n e_i$ represents how the error signals e_i are sent backward to update the feed-forward weights. With the use of random backpropagation weights, the error is instead propagated backward as $\delta_{\text{FA}}(h) = \frac{1}{\sqrt{p}} b \sum_{i=1}^n e_i$.

Lillicrap et al. (2016) notice a decreasing angle between $\delta_{\text{BP}}(h)$ and $\delta_{\text{FA}}(h)$ during training, which is a sufficient condition to ensure that the algorithm converges. In the case of k-way classification, the last layer has k nodes, β and b are $p \times k$ matrices, and each error term e_i is a k-vector. In the regression setting, $k = 1$ so the angle between $\delta_{\text{BP}}(h)$ and $\delta_{\text{FA}}(h)$ is the same as the angle between β and b. Intuitively, the possibility for alignment is seen in the fact that while the updates for W use the error weighted by the random weights b, the updates for β indirectly involve W, allowing for the possibility that dependence on b will be introduced into β.

Our first result shows that, in fact, alignment will *not* occur in the over-parameterized setting. (So, while the error may still converge, "feedback alignment" may be a bit of a misnomer for the algorithm.)

- The cosine of the angle between the p-dimensional vectors δ_{FA} and δ_{BP} satisfies $\cos\angle(\delta_{\text{FA}}, \delta_{\text{BP}}(t)) = \cos\angle(b, \beta(t)) = O\big(\frac{n}{\sqrt{p}}\big)$.

However, we show that regularizing the parameters will cause δ_{BP} to align with δ_{FA} and therefore the parameters β to align with b. Since $\beta(0)$ and b are high dimensional Gaussian vectors, they are nearly orthogonal with high probability. The effect of regularization can be seen as shrinking the component of $\beta(0)$ in the parameters over time. Our next result establishes this precisely in the linear case.

- Supposing that $\psi(u) = u$, then introducing a ridge penalty $\lambda(t)\|\beta\|^2$ where $\lambda(t) = \lambda$ for $t \leq T$ and $\lambda(t) = 0$ for $t > T$ on β causes the parameters to align, with $\cos\angle(b, \beta(t)) \geq c > 0$ for sufficiently large t.

The technical conditions are given in Theorem 2.4.6. Our simulations are consistent with this result, and also show alignment with a constant regularization $\lambda(t) \equiv \lambda$, for both linear and nonlinear activation functions. Finally, we complement this result by showing that convergence is preserved with regularization, for general activation functions. This is presented in Theorem 2.4.2.

2.3 Convergence with Random Backpropagation Weights

Due to the replacement of backward weights with the random backpropagation weights, there is no guarantee *a priori* that the algorithm will reduce the squared error loss $\mathcal{L}$. Lillicrap et al. (2020) study the convergence on two-layer linear networks in a continuous time setting. Through the analysis of a system of differential equations on the network parameters, convergence to the true linear target function is shown, in the population setting of arbitrarily large training data. Among recent studies of over-parametrized networks under backpropagation, the neural tangent kernel (NTK) is heavily utilized to describe the evolution of the network during training (Jacot et al., 2018). For any neural network $f(x, \theta)$ with parameter θ, the NTK is defined as

$$K_f(x, y) = \left\langle \frac{\partial f(x, \theta)}{\partial \theta}, \frac{\partial f(y, \theta)}{\partial \theta} \right\rangle.$$

Given a dataset $\{(x_i, y_i)\}_{i=1}^n$, we can also consider its corresponding Gram matrix $K = (K_f(x_i, x_j))_{n \times n}$. Jacot et al. (2018) show that in the infinite width limit, K_f converges to a constant at initialization and does not drift away from initialization throughout training. In the over-parameterized setting, if the Gram matrix K is positive definite, then K will remain close to its initialization during training, resulting in linear convergence of the squared error loss (Du et al., 2018b, 2019; Gao and

Lafferty, 2020). For the two-layer network $f(x,\theta)$ defined in (2.1) with $\theta = (\beta, W)$, the kernel K_f can be written in two parts, G_f and H_f, which correspond to β and W respectively:

$$K_f(x,y) = G_f(x,y) + H_f(x,y) \coloneqq \left\langle \frac{\partial f(x,\theta)}{\partial \beta}, \frac{\partial f(y,\theta)}{\partial \beta} \right\rangle + \sum_{r=1}^{p} \left\langle \frac{\partial f(x,\theta)}{\partial w_r}, \frac{\partial f(y,\theta)}{\partial w_r} \right\rangle.$$

Under the feedback alignment scheme with random backward weights b, G_f remains the same as for standard backpropagation, while one of the gradient terms $\frac{\partial f}{\partial w_r}$ in H_f changes to

$$\widetilde{\frac{\partial f(x,\theta)}{\partial w_r}} = \frac{1}{\sqrt{p}} b_r \psi'(w_r^\top x) x,$$

with H_f replaced by

$$H_f = \sum_{r=1}^{p} \left\langle \widetilde{\frac{\partial f(x,\theta)}{\partial w_r}}, \frac{\partial f(y,\theta)}{\partial w_r} \right\rangle.$$

As a result, H_f is no longer positive semi-definite and close to 0 at initialization if the network is over-parameterized. However, if $G = (G_f(x_i,x_j))_{n\times n}$ is positive definite and $H = (H_f(x_i,x_j))_{n\times n}$ remains small during training, we are still able to show that the loss $\mathcal{L}$ will converge to zero exponentially fast.

Assumption 2.3.1. *Define the matrix $\overline{G} \in \mathbb{R}^{n\times n}$ with entries*

$$\overline{G}_{i,j} = \mathbb{E}_{w\sim\mathcal{N}(0,I_p)} \psi(w^\top x_i)\psi(w^\top x_j).$$

We assume that the minimum eigenvalue satisfies $\lambda_{\min}(\overline{G}) \geq \gamma$, where γ is a positive constant.

Theorem 2.3.2. *Let $W(0)$, $\beta(0)$ and b have i.i.d. standard Gaussian entries. Assume*

1. *Assumption 2.3.1 holds,*

2. *ψ is smooth, ψ, ψ' and ψ'' are bounded,*

3. $|y_i|$ and $\|x_i\|$ are bounded for all $i \in [n]$.

Then there exists positive constants c_1, c_2, C_1 and C_2, such that for any $\delta \in (0,1)$, if

$$p \geq \max\left(C_1 \frac{n^2}{\delta\gamma^2}, C_2 \frac{n^4 \log p}{\gamma^4}\right),$$

then with probability at least $1-\delta$ we have that

$$\|e(t+1)\| \leq (1 - \frac{\eta\gamma}{4})\|e(t)\| \tag{2.4}$$

and

$$\|w_r(t) - w_r(0)\| \leq c_1 \frac{n\sqrt{\log p}}{\gamma\sqrt{p}}, \quad |\beta_r(t) - \beta_r(0)| \leq c_2 \frac{n}{\gamma\sqrt{p}} \tag{2.5}$$

for all $r \in [p]$ and $t > 0$.

We note that the matrix $\overline{G}$ in Assumption 2.3.1 is the expectation of G with respect to the random initialization, and is thus close to $\overline{G}$ due to concentration. To justify the assumption, we provide the following proposition, which states that Assumption 2.3.1 holds when the inputs x_i are drawn independently from a Gaussian distribution. The proofs of Theorem 2.3.2 and Proposition 2.3.3 are deferred to Section 2.5.2.

Proposition 2.3.3. *Suppose $x_1, ..., x_n \overset{i.i.d.}{\sim} \mathcal{N}(0, I_d/d)$ and the activation function ψ is sigmoid or tanh. If $d = \Omega(n)$, then Assumption 2.3.1 holds with high probability.*

2.4 Alignment with Random Backpropagation Weights

The most prominent characteristic of the feedback alignment algorithm is the phenomenon that the error signals propagated with the forward weights align with those propagated with fixed random backward weights during training. Specifically, if we denote $h \in \mathbb{R}^p$ to be the hidden layer of the network, then we write $\delta_{\text{BP}}(h) := \frac{\partial \mathcal{L}}{\partial h}$ to represent the error signals with respect to the hidden layer that are backpropagated with the feed-forward weights and $\delta_{\mathrm{FA}}(h)$ as the error signals computed with fixed random backward weights. In particular, the error signals $\delta_{\mathrm{BP}}(h)$ and $\delta_{\mathrm{FA}}(h)$ for the two-layer network (2.1) are given by

$$\delta_{\mathrm{BP}}(h) = \frac{1}{\sqrt{p}}\beta\sum_{i=1}^{n} e_i \quad \text{and} \quad \delta_{\mathrm{FA}}(h) = \frac{1}{\sqrt{p}}b\sum_{i=1}^{n} e_i.$$

Lillicrap et al. (2016) notice a decreasing angle between $\delta_{\mathrm{BP}}(h)$ and $\delta_{\mathrm{FA}}(h)$ during training. We formalize this concept of alignment by the following definition.

Definition 2.4.1. *We say a two-layer network aligns with the random weights b during training if there exists a constant $c > 0$ and time T_c such that for all $t > T_c$,*

$$\cos\angle(\delta_{\mathrm{FA}}, \delta_{\mathrm{BP}}(t)) = \cos\angle(b, \beta(t)) = \frac{\langle b, \beta(t)\rangle}{\|b\|\|\beta(t)\|} \geq c.$$

2.4.1 Regularized feedback alignment

Unfortunately, alignment between $\beta(t)$ and b is not guaranteed for over-parameterized networks and the loss (2.2). In particular, we control the cosine value of the angle by inequalities (2.5) from Theorem 2.3.2, *i.e.*,

$$\left|\cos\angle(b, \beta(t))\right| \leq \frac{|\langle \frac{b}{\|b\|}, \beta(0)\rangle| + \|\beta(t) - \beta(0)\|}{\|\beta(0)\| - \|\beta(t) - \beta(0)\|} = O\left(\frac{n}{\sqrt{p}}\right),$$

which indicates that $\beta(t)$ and b become orthogonal as the network becomes wider. Intuitively, this can be understood as resulting from the parameters staying near their initializations during training when p is large, where $\beta(0)$ and b are almost orthogonal to each other. This motivates us to regularize the network parameters. We consider in this work the squared error loss with an ℓ_2 regularization term on β:

$$\mathcal{L}(t, W, \beta) = \frac{1}{2}\sum_{i=1}^{n} \big(f(x_i) - y_i\big)^2 + \frac{1}{2}\lambda(t)\|\beta\|^2, \tag{2.6}$$

where $\{\lambda(t)\}_{t=0}^{\infty}$ is a sequence of regularization rates, which defines a series of loss functions for different training steps t. Thus, the update for w_r remains the same and the update for β changes to

$$\beta_r(t+1) = (1-\lambda(t))\beta_r(t) - \frac{\eta}{\sqrt{p}}\sum_{i=1}^{n} e_i(t)\psi(w_r(t)^\top x_i), \quad \text{for } r \in [p].$$

Comparing to Algorithm 1, an extra contraction factor $1-\lambda(t)$ is added in the update of $\beta(t)$, which doesn't affect the locality of the algorithm but helps the alignment by shrinking the component of $\beta(0)$ in $\beta(t)$.

Algorithm 2 Regularized Feedback Alignment on Two-Layer Networks

```
Input: Dataset $\{(x_i, y_i)\}_{i=1}^n$, and step size $\eta$, and regularization $\{\lambda(t)\}$.
1: initialize $W(0)$, $\beta(0)$ and $b$ as Gaussian
2: for $t = 0, 1, 2, \ldots$ do
3:     $\beta_r(t+1) = (1-\lambda(t))\beta_r(t) - \frac{\eta}{\sqrt{p}}\sum_{i=1}^n e_i(t)\psi(w_r(t)^\top x_i)$
4:     $w_r(t+1) = w_r(t) - \frac{\eta}{\sqrt{p}}\sum_{i=1}^n e_i(t)b_r\psi'(w_r(t)^\top x_i)x_i$
5:     for $r \in [p]$
6: end for
```

Following Theorem 2.3.2, we provide an error bound for regularized feedback alignment in Theorem 2.4.2. Since regularization terms $\lambda(t)$ make additional contributions to the error $e(t)$ as well as to the kernel matrix G, an upper bound on $\sum_{t\geq 0}\lambda(t)$ is needed to ensure positivity of the minimal eigenvalue of G during training, in order for the error $e(t)$ to be controlled. In particular, if there is no regularization, *i.e.*, $\lambda(t) = 0$ for all $t \geq 0$, then we recover exponential convergence for the error $e(t)$ as in Theorem 2.3.2. The proof of Theorem 2.4.2 is also deferred to Section 2.5.2.

Theorem 2.4.2. *Assume all the conditions from Theorem 2.3.2. Assume* $\sum_{t=0}^{\infty} \lambda(t) \leq \tilde{S}_\lambda = \tilde{c}_S \frac{\gamma^2 \sqrt{p}}{\eta n^2 \sqrt{\log p}}$ *for some constant* $\tilde{c}_S$*. Then there exist positive constants* C_1 *and* C_2*, such that for any* $\delta \in (0,1)$*, if* $p \geq \max\big(C_1 \frac{n^2}{\delta \gamma^2}, C_2 \frac{n^4 \log p}{\gamma^4}\big)$*, then with probability at least* $1-\delta$*, we have*

$$\|e(t+1)\| \leq \Big(1 - \frac{\eta\gamma}{4} - \eta\lambda(t)\Big)\|e(t)\| + \lambda(t)\|y\| \tag{2.7}$$

for all $t \geq 0$.

2.4.2 Alignment analysis for linear networks

In this section, we focus on the theoretical analysis of alignment for linear networks, which is equivalent to setting the activation function ψ to the identity map. The loss function can be written as

$$\mathcal{L}(t, W, \beta) = \frac{1}{2}\Big\|\frac{1}{\sqrt{p}} X W^\top \beta - y\Big\|^2 + \frac{\lambda(t)}{2}\|\beta\|^2,$$

where $X = (x_1, \ldots, x_n)^\top$; this is a form of over-parameterized ridge regression. Before presenting our results on alignment, we first provide a linear version of Theorem 2.4.2 that adopts slightly different conditions.

Theorem 2.4.3. *Assume* (1) $\|y\| = \Theta(\sqrt{n})$, $\lambda_{\min}(XX^\top) > \gamma$ *and* $\lambda_{\max}(XX^\top) < M$ *for some constants* $M > \gamma > 0$*, and* (2) $\sum_{t=0}^{\infty} \lambda(t) \leq S_\lambda = c_S \frac{\gamma\sqrt{\gamma p}}{\eta \sqrt{n} M}$ *for some constant* c_S*. Then for any* $\delta \in (0,1)$*, if* $p = \Omega(\frac{Md\log(d/\delta)}{\gamma})$*, the following inequality holds for all* $t \geq 0$ *with probability at least* $1-\delta$*:*

$$\|e(t+1)\| \leq \big(1 - \frac{\eta\gamma}{2} - \eta\lambda(t)\big)\|e(t)\| + \lambda(t)\|y\|. \tag{2.8}$$

We remark that in the linear case, the kernel matrix G reduces to the form

$XW^\top WX^\top$ and its expectation $\overline{G}$ at initialization also reduces to $XX^\top$. Thus, Assumption 2.3.1 holds if $XX^\top$ is positive definite, which is equivalent to the x_i's being linearly independent. The result of Theorem 2.4.2 can not be directly applied to the linear case since we assume that ψ is bounded, which is true for sigmoid or tanh but not for the identity map. This results in a slightly different order for S_λ and an improved order for p.

Our results on alignment also rely on an isometric condition on X, which requires the minimum and the maximum eigenvalues of $XX^\top$ to be sufficiently close (*cf.* Definition 2.4.4). On the other hand, this condition is relatively mild and can be satisfied when X has random Gaussian entries with a gentle dimensional constraint, as demonstrated by Proposition 2.4.5. Finally, we show in Theorem 2.4.6 that under a simple regularization strategy where a constant regularization is adopted until a cutoff time T, regularized feedback alignment achieves alignment if X satisfies the isometric condition.

Definition 2.4.4 ((γ, ε)-Isometry). *Given positive constants γ and ε, we say X is (γ, ε)-isometric if $\lambda_{\min}(XX^\top) \geq \gamma$ and $\lambda_{\max}(XX^\top) \leq (1+\varepsilon)\gamma$.*

Proposition 2.4.5. *Assume $X \in \mathbb{R}^{n\times d}$ has independent entries drawn from $N(0, 1/d)$. For any $\varepsilon \in (0, 1/2)$ and $\delta \in (0,1)$, if $d = \Omega(\frac{1}{\varepsilon}\log\frac{n}{\delta} + \frac{n}{\varepsilon}\log\frac{1}{\varepsilon})$, then X is $(1-\varepsilon, 4\varepsilon)$-isometric with probability $1-\delta$.*

Theorem 2.4.6. *Assume all conditions from Theorem 2.4.3 hold and X is (γ, ε)-isometric with a small constant ε. Let the regularization weights satisfy*

$$\lambda(t) = \begin{cases} \lambda, & t \leq T, \\ 0, & t > T, \end{cases}$$

with $\lambda = L\gamma$ and $T = \lfloor S_\lambda/\lambda \rfloor$ for some large constant L. Then for any $\delta \in (0,1)$, if $p = \Omega(d\log(d/\delta))$, with probability at least $1-\delta$, regularized feedback alignment *achieves alignment. Specifically, there exist a positive constant $c = c_\delta$ and time T_c, such that $\cos\angle(b, \beta(t)) \geq c$ for all $t > T_c$.*

We defer the proofs of Proposition 2.4.5, Theorem 2.4.3 and Theorem 2.4.6 to Section 2.5.3. In fact, we prove Theorem 2.4.6 by directly computing $\beta(t)$ and the cosine of the angle. Although b doesn't show up in the update of β, it can still propagate to β through W. Since the size of the component of b in $\beta(t)$ depends on the inner-product $\langle e(t), e(t')\rangle$ for all previous steps $t' \leq t$, the norm bound (2.8) from Theorem 2.4.3 is insufficient; thus, a more careful analysis of $e(t)$ is required.

We should point out that the constant c in the lower bound is independent of the sample size n, input dimension d, network width p and learning rate η. We also remark that the cutoff schedule of $\lambda(t)$ is just chosen for simplicity. For other schedules such as inverse-squared decay or exponential decay, one could also obtain the same alignment result as long as the summation of $\lambda(t)$ is less than S_λ.

Large sample scenario. In Theorems 2.4.3 and 2.4.6, we consider the case where the sample size n is less than the input dimension d, so that positive definiteness of $XX^\top$ can be established. However, both results still hold for $n > d$. In fact, the squared error loss $\mathcal{L}$ can be written as

$$\sum_{i=1}^{n} \left(f(x_i) - y\right)^2 = \left\|\frac{1}{\sqrt{p}} XW^\top \beta - y\right\|^2 = \left\|\frac{1}{\sqrt{p}} XW^\top \beta - \bar{y}\right\|^2 + \|\bar{y} - y\|^2,$$

where $\bar{y}$ denotes the projection of y onto the column space of X. Without loss of generality, we assume $y = \bar{y}$. As a result, y and the columns of X are all in the same d-dimensional subspace of $\mathbb{R}^n$ and $XX^\top$ is positive definite on this subspace, as long as X has full column rank. Consequently, we can either work on this subspace of $\mathbb{R}^n$ or project all the vectors onto $\mathbb{R}^d$, and the isometric condition is revised to only consider the d nonzero eigenvalues of $XX^\top$.

2.5 Proofs for Technical Results

In this section, we provide proofs for the theoretical results in Section 2.5.2 and Section 2.4. To help make the proof more readable, we use c, C to denote the global constants whose values may vary from line to line.

2.5.1 Technical Lemmas

To begin with, we list technical lemmas that will be used in the proofs, with references. The first is a variant of the Restricted Isometry Property that bounds the spectral norm of a random Gaussian matrix around 1 with high probability.

Lemma 2.5.1 (Hand and Voroninski, 2019). *Let $A \in \mathbb{R}^{m\times n}$ has i.i.d. $\mathcal{N}(0, 1/m)$ entries. Fix $0 < \varepsilon < 1$, $k < m$, and a subspace $T \subseteq \mathbb{R}^n$ of dimension k, then there exists universal constants c_1 and γ_1, such that with probability at least $1 - (c_1/\varepsilon)^k e^{-\gamma_1 \varepsilon m}$,*

$$(1-\varepsilon)\|v\|_2^2 \leq \|Av\|_2^2 \leq (1+\varepsilon)\|v\|_2^2, \quad \forall v \in T.$$

Let us take $k = n$ in Lemma 2.5.1 to get the following corollary.

Corollary 2.5.2. *Let $A \in \mathbb{R}^{m\times n}$ has i.i.d. $\mathcal{N}(0, 1/m)$ entries. For any $0 < \varepsilon < 1$, there exists universal constants c_2 and γ_2, such that with probability at least $1 - (c_2/\varepsilon)^d e^{-\gamma_2 \varepsilon m}$,*

$$\|A^\top A - I_m\| \leq \varepsilon$$

Then following lemma gives tail bounds for χ^2 random variables.

Lemma 2.5.3 (Laurent and Massart, 2000). *Suppose $X \sim \chi^2_p$, then for all $t \geq 0$ it* *holds*

$$\mathbb{P}\{X - p \geq 2\sqrt{pt} + 2t\} \leq e^{-t}$$

and

$$\mathbb{P}\{X - p \leq -2\sqrt{pt}\} \leq e^{-t}.$$

For two independent random Gaussian vectors, their inner product can be controlled with the following tail bound.

Lemma 2.5.4 (Gao and Lafferty, 2020)**.** *Let $X, Y \in \mathbb{R}^p$ be independent random Gaussian vectors where $X_r \sim \mathcal{N}(0,1)$ and $Y_r \sim \mathcal{N}(0,1)$ for all $r \in [p]$, then it holds*

$$\mathbb{P}(|X^\top Y| \geq \sqrt{2pt} + 2t) \leq 2e^t.$$

2.5.2 Convergence on Two-Layer Nonlinear Networks

We consider the family of neural networks

$$f(x) = \frac{1}{\sqrt{p}} \sum_{r=1}^{p} \beta_r \psi(w_r^\top x) = \frac{1}{\sqrt{p}} \beta^\top \psi(Wx) \tag{2.9}$$

where $\beta \in \mathbb{R}^p$, $W = (w_1, ..., w_p)^\top \in \mathbb{R}^{p \times d}$, and ψ is an activation function. Given data, the loss function is

$$\mathcal{L}(W, \beta) = \frac{1}{2} \sum_{i=1}^{n} (f(x_i) - y_i)^2 = \frac{1}{2} \sum_{i=1}^{n} \Big(\frac{1}{\sqrt{p}} \beta^\top \psi(W x_i) - y \Big)^2. \tag{2.10}$$

The feedback alignment algorithm has updates

$$\begin{aligned} W(t+1) &= W(t) - \eta \frac{1}{\sqrt{p}} \sum_{i=1}^{n} D_i(t) b x_i^\top e_i(t) \\ \beta(t+1) &= \beta(t) - \eta \frac{1}{\sqrt{p}} \sum_{i=1}^{n} \psi(W(t)x_i) e_i(t) \end{aligned} \tag{2.11}$$

where $D_i(t) = \operatorname{diag}(\psi'(W(t)x_i))$ and $e_i(t) = \frac{1}{\sqrt{p}}\beta(t)^\top \psi(W(t)x_i) - y_i$.

Concentration Results

Lemma 2.5.5 (Lemma A.7 in Gao and Lafferty, 2020)**.** *Assume $x_1, ..., x_n \overset{i.i.d.}{\sim} \mathcal{N}(0, I_d/d)$. We define matrix $\widetilde{G} \in \mathbb{R}^{n\times n}$ with entries*

$$\widetilde{G}_{i,j} = |\mathbb{E}\psi'(Z)|^2 \frac{x_i^\top x_j}{\|x_i\|\|x_j\|} + (\mathbb{E}|\psi(Z)|^2 - |\mathbb{E}\psi'(Z)|^2)\mathbb{I}\{i=j\}$$

where $Z \sim \mathcal{N}(0,1)$. If $d = \Omega(\log n)$, then with high probability, we have

$$\|\overline{G} - \widetilde{G}\|^2 \lesssim \frac{\log n}{d} + \frac{n^2}{d^2}.$$

Proof of Proposition 2.3.3. If ψ is sigmoid or tanh, for a standard Gaussian random variable Z, we have

$$\gamma \coloneqq \frac{1}{2}(\mathbb{E}|\psi(Z)|^2 - |\mathbb{E}\psi'(Z)|^2) > 0.$$

From Lemma 2.5.5, we know that with high probability $\lambda_{\min}(\overline{G}) \geq \lambda_{\min}(\widetilde{G}) - \|\overline{G} - \widetilde{G}\| \geq 2\gamma - C(\sqrt{\frac{\log n}{d}} + \frac{n}{d}) \geq \gamma$. □

Lemma 2.5.6. *Assume $W(0)$, $\beta(0)$ and b have i.i.d. standard Gaussian entries. Given $\delta \in (0,1)$, if $p = \Omega(n/\delta)$, then with probability $1-\delta$*

$$\frac{1}{p}\sum_{r=1}^{p} |b_r| \leq c, \tag{2.12}$$

$$\frac{1}{p}\sum_{r=1}^{p}|b_r\beta_r(0)| \leq c, \tag{2.13}$$

$$\|e(0)\| \leq c\sqrt{n}, \tag{2.14}$$

$$\max_{r\in[p]}|b_r| \leq 2\sqrt{\log p}. \tag{2.15}$$

Proof. We will show each inequality holds with probability at least $1-\frac{\delta}{4}$, then by a union bound, all of them hold with probability at least $1-\delta$. Since $\mathbb{V}\mathrm{ar}(\frac{1}{p}\sum_{r=1}^{p}|b_r|) \leq \frac{\mathbb{V}\mathrm{ar}(|b_0|)}{p}$, by Chebyshev's inequality, we have

$$\mathbb{P}(\frac{1}{p}\sum_{r=1}^{p}|b_r| > \mathbb{E}(b_1)+1) \leq \frac{\mathbb{V}\mathrm{ar}(|b_1|)}{p} \leq \delta/4$$

if $p \geq 4\mathbb{V}\mathrm{ar}(|b_1|)/\delta$, which gives (2.12). The proof for (2.13) is similar since $\mathbb{V}\mathrm{ar}(\frac{1}{p}\sum_{r=1}^{p}|b_r\beta_r(0)|) = O(1/p)$. To prove (2.14), since $|y_i|$ and $\|x_i\|$ are bounded, it suffices to show $|u_i(0)| \leq c$ for all $i \in [n]$. Actually, by independence, we have

$$\mathbb{V}\mathrm{ar}(u_i(0)) = \mathbb{V}\mathrm{ar}\Big(\frac{1}{p}\sum_{r=1}^{p}\beta_r(0)\psi(w_r(0)^\top x_i)\Big) = \frac{1}{p}\mathbb{V}\mathrm{ar}\Big(\beta_1(0)\psi(w_1(0)^\top x_i)\Big) = O(1/p).$$

By Chebyshev's inequality, we have for each $i \in [n]$

$$\mathbb{P}(|u_i(0)| > c) \leq \frac{\mathbb{V}\mathrm{ar}(u_i(0))}{c^2} \leq \frac{\delta}{4n}$$

where we require $p = \Omega(n/\delta)$. With a union bound argument, we can show (2.14). Finally, (2.15) followed from standard Gaussian tail bounds and union bound argument, yielding

$$\mathbb{P}(\max_{r\in[p]} |b_r| > 2\sqrt{\log p}) \leq \sum_{r\in[p]} \mathbb{P}(|b_r| > 2\sqrt{\log p}) \leq 2pe^{-2\log p} = \frac{2}{p} \leq \frac{\delta}{4}.$$

□

Lemma 2.5.7. *Under the conditions of Theorem 2.3.2, we define matrices* $G(0), H(0) \in \mathbb{R}^{n\times n}$ *with entries*

$$G_{ij}(0) = \frac{1}{p}\psi(W(0)x_i)^\top \psi(W(0)x_j) = \frac{1}{p}\sum_{r=1}^{p} \psi(w_r(0)^\top x_i)\psi(w_r(0)^\top x_j) \tag{2.16}$$

and

$$H_{ij}(0) = \frac{x_i^\top x_j}{p}\beta(0)^\top D_i(0)D_j(0)b = \frac{1}{p}\sum_{r=1}^{p} \beta_r(0)b_r\psi'(w_r(0)^\top x_i)\psi'(w_r(0)^\top x_j). \tag{2.17}$$

For any $\delta \in (0,1)$, *if* $p = \Omega(\frac{n^2}{\delta\gamma^2})$, *then with probability at least* $1-\delta$, *we have* $\lambda_{\min}(G(0)) \geq \frac{3}{4}\gamma$ *and* $\|H(0)\| \leq \frac{\gamma}{4}$.

Proof. By independence and boundedness of ψ and ψ', we have $\mathbb{V}\text{ar}(G_{ij}(0)) = O(1/p)$ and $\mathbb{V}\text{ar}(H_{ij}(0)) = O(1/p)$. Since $\mathbb{E}(G(0)) = \overline{G}$, we have

$$\mathbb{E}\|G(0) - \overline{G}\|^2 \leq \mathbb{E}\|G(0) - \overline{G}\|_F^2 = O(\frac{n^2}{p}).$$

By Markov's inequality, when $p = \Omega(\frac{n^2}{\delta\gamma^2})$

$$\mathbb{P}(\|G(0) - \overline{G}\| > \frac{\gamma}{4}) \leq O(\frac{n^2}{p\gamma^2}) \leq \frac{\delta}{2}.$$

Similarly we have $\mathbb{P}(\|H(0)\| > \frac{\gamma}{4}) \leq \frac{\delta}{2}$, since $\mathbb{E}(H(0)) = 0$. Then with probability at least $1-\delta$, $\lambda_{\min}(G(0)) \geq \lambda_{\min}(\overline{G}) - \gamma/4 \geq \frac{3}{4}\gamma$, and $\|H(0)\| \leq \gamma/4$. □

Proof of Theorem 2.3.2

Lemma 2.5.8. *Assume all the inequalities from Lemma 2.5.6 hold. Under the conditions of Theorem 2.3.2, if the error bound (2.4) holds for all $t = 1, 2, ..., t' - 1$, then the bounds (2.5) hold for all $t \leq t'$.*

Proof. From the feedback alignment updates (2.11), we have for all $t \leq T$

$$\begin{aligned}
|\beta_r(t) - \beta_r(0)| &\leq \frac{\eta}{\sqrt{p}} \sum_{s=0}^{t-1} \sum_{i=1}^{n} |\psi(w_r(t)x_i)e_i(t)| \\
&\leq c\frac{\eta}{\sqrt{p}} \sum_{s=0}^{t-1} \sum_{i=1}^{n} |e_i(t)| \\
&\leq c\frac{\eta\sqrt{n}}{\sqrt{p}} \sum_{s=0}^{t-1} \|e(t)\| \\
&\leq c\frac{\eta\sqrt{n}}{\sqrt{p}} \sum_{s=0}^{t-1} (1 - \frac{\gamma\eta}{4})^t \|e(0)\| \\
&\leq c\frac{\sqrt{n}}{\gamma\sqrt{p}} \|e(0)\| \\
&\leq c\frac{n}{\gamma\sqrt{p}}
\end{aligned}$$

where we use the fact that ψ is bounded and (2.14). We also have

$$\begin{aligned}
\|w_r(t) - w_r(0)\| &\leq \frac{\eta}{\sqrt{p}} \sum_{s=0}^{t-1} \sum_{i=1}^{n} \|\psi'(w_r(t)^\top x_i) b_r x_i e_i(t)\| \\
&\leq c\frac{\eta}{\sqrt{p}} \sum_{s=0}^{t-1} \sum_{i=1}^{n} |b_r||e_i(t)| \\
&\leq c|b_r|\frac{\eta\sqrt{n}}{\sqrt{p}} \sum_{s=0}^{t-1} \|e(t)\| \\
&\leq c|b_r|\frac{\sqrt{n}}{\gamma\sqrt{p}} \|e(0)\| \\
&\leq c\frac{n\sqrt{\log p}}{\gamma\sqrt{p}}
\end{aligned}$$

where we use that ψ' is bounded, (2.14) and (2.15). □

Lemma 2.5.9. *Assume all the inequalities from Lemma 2.5.6 hold. Under the conditions of Theorem 2.3.2, if the bound for the weights difference (2.5) holds for all $t \leq t'$ and error bound (2.4) holds for all $t \leq t' - 1$, then (2.4) holds for $t = t'$.*

Proof. We start with analyzing the error $e(t)$ according to

$$
\begin{aligned}
e_i(t+1) &= \frac{1}{\sqrt{p}}\beta(t+1)^\top \psi(W(t+1)x_i) - y_i \\
&= \frac{1}{\sqrt{p}}\beta(t+1)^\top (\psi(W(t+1)x_i) - \psi(W(t)x_i)) \\
&\quad + \frac{1}{\sqrt{p}}(\beta(t+1) - \beta(t))^\top \psi(W(t)x_i) \\
&\quad + \frac{1}{\sqrt{p}}\beta(t)^\top \psi(W(t)x_i) - y_i \\
&= e_i(t) - \frac{\eta}{p}\beta(t+1)^\top D_i(t) \sum_{j=1}^{n} D_j(t) b x_j^\top x_i e_j(t) \\
&\quad - \frac{\eta}{p}\sum_{j=1}^{n} \psi(W(t)x_j)^\top \psi(W(t)x_i) e_j(t) + v_i(t) \\
&= e_i(t) - \eta \sum_{j=1}^{n} \big(H_{ij}(t) + G_{ij}(t)\big) e_j(t) + v_i(t)
\end{aligned}
$$

where

$$
\begin{aligned}
G_{ij}(t) &= \frac{1}{p}\psi(W(t)x_j)^\top \psi(W(t)x_i) \\
H_{ij}(t) &= \frac{x_i^\top x_j}{p}\beta(t+1)^\top D_i(t) D_j(t) b
\end{aligned}
$$

and $v_i(t)$ is the residual term from the Taylor expansion

$$
v_i(t) = \frac{1}{2\sqrt{p}} \sum_{r=1}^{p} \beta_r(t+1) |(w_r(t+1) - w_r(t))^\top x_i|^2 \psi''(\xi_{ri}(t))
$$

with $\xi_{ri}(t)$ between $w_r(t)^\top x_i$ and $w_r(t+1)^\top x_i$. We can also rewrite the above iteration in vector form as

$$
e(t+1) = e(t) - \eta(G(t) + H(t))e(t) + v(t). \tag{2.18}
$$

Now for $t = t' - 1$, we wish to show that both $G(t)$ and $H(t)$ are close to their initialization. Notice that

$$\begin{aligned}
|G_{ij}(t) - G_{ij}(0)| &= \frac{1}{p}\Big|\psi(W(t)x_j)^\top \psi(W(t)x_i) - \psi(W(t)x_j)^\top \psi(W(t)x_i)\Big| \\
&\leq \frac{1}{p}\sum_{r=1}^{p} |\psi(w_r(t)^\top x_j)||\psi(w_r(t)^\top x_i) - \psi(w_r(0)^\top x_i)| \\
&\quad + \frac{1}{p}\sum_{r=1}^{p} |\psi(w_r(0)^\top x_i)||\psi(w_r(t)^\top x_j) - \psi(w_r(0)^\top x_j)| \\
&\leq c\frac{1}{p}\sum_{r=1}^{p} |w_r(t)^\top x_i - w_r(0)^\top x_i| + \frac{1}{p}\sum_{r=1}^{p} |w_r(t)^\top x_j - w_r(0)^\top x_j| \\
&\leq c_0 \frac{n\sqrt{\log p}}{\gamma\sqrt{p}}(\|x_i\| + \|x_j\|)
\end{aligned}$$

where the second inequality is due to the boundedness of ψ and ψ', and the last inequality is by (2.5). Then we have

$$\|G(t) - G(0)\| \leq \max_{j\in[n]} \sum_{i=1}^{n} |G_{ij}(t) - G_{ij}(0)| \leq c_0 \frac{n^2\sqrt{\log p}}{\gamma\sqrt{p}}. \tag{2.19}$$

For matrix $H(t)$, we similarly have

$$\begin{aligned}
|H_{ij}(t) - H_{ij}(0)| &\le \frac{|x_i^\top x_j|}{p}\Big|\beta(t+1)^\top D_i(t)D_j(t)b - \beta(0)^\top D_i(0)D_j(0)b\Big| \\
&\le \frac{\|x_i\|\|x_j\|}{p}\sum_{r=1}^{p}\Big|b_r\beta_r(t+1)\psi'(w_r(t)^\top x_i)\psi'(w_r(t)^\top x_j) \\
&\quad - b_r\beta_r(0)\psi'(w_r(0)^\top x_i)\psi'(w_r(0)^\top x_j)\Big| \\
&\le \frac{|\|x_i\|\|x_j\||}{p}\sum_{r=1}^{p}\Big(|b_r||\beta_r(t+1)-\beta_r(0)||\psi'(w_r(t)^\top x_i)\psi'(w_r(t)^\top x_j)| \\
&\quad + |b_r||\beta_r(0)||\psi'(w_r(t)^\top x_i) - \psi'(w_r(0)^\top x_i)||\psi'(w_r(t)^\top x_j)| \\
&\quad + |b_r||\beta_r(0)||\psi'(w_r(0)^\top x_i)||\psi'(w_r(t)^\top x_j) - \psi'(w_r(0)^\top x_j)|\Big) \\
&\le c\frac{\|x_i\|\|x_j\|}{p}\sum_{r=1}^{p}\Big(|b_r|\frac{n}{\gamma\sqrt{p}} + |b_r||\beta_r(0)|\frac{n\sqrt{\log p}}{\gamma\sqrt{p}}(\|x_i\|+\|x_j\|)\Big) \\
&\le c_1\frac{n}{\gamma\sqrt{p}} + c_2\frac{n\sqrt{\log p}}{\gamma\sqrt{p}}.
\end{aligned}$$

It follows that

$$\|H(t)-H(0)\| \le \max_{j\in[n]}\sum_{i=1}^{n}|H_{ij}(t)-H_{ij}(0)| \le c_1\frac{n^2}{\gamma\sqrt{p}} + c_2\frac{n^2\sqrt{\log p}}{\gamma\sqrt{p}}. \tag{2.20}$$

Next, we bound the residual term $v_i(t)$. Since ψ'' is bounded, we have

$$\begin{aligned}
|v_i(t)| &\le c\frac{1}{\sqrt{p}}\sum_{r=1}^{p}|\beta_r(t+1)|\|w_r(t+1)-w_r(t)\|^2 \\
&\le c\frac{1}{\sqrt{p}}\frac{\eta^2}{p}\sum_{r=1}^{p}|\beta_r(t+1)|\Big(\sum_{i=1}^{n}\|\psi'(w_r(t)^\top x_i)b_r x_i e_i(t)\|\Big)^2 \\
&\le c\frac{1}{\sqrt{p}}\frac{\eta^2}{p}\sum_{r=1}^{p}|\beta_r(t+1)||b_r|^2\Big(\sum_{i=1}^{n}|e_i(t)|\Big)^2 \\
&\le c\frac{\eta^2 n}{\sqrt{p}}\|e(t)\|^2 \\
&\le c_3\frac{\eta^2 n\sqrt{n}}{\sqrt{p}}\|e(t)\|.
\end{aligned}$$

This leads to the bound

$$\|v(t)\| = \Big(\sum_{i=1}^{n} |v_i(t)|^2\Big)^{1/2} \le c_3 \frac{\eta^2 n^2}{\sqrt{p}} \|e(t)\|. \tag{2.21}$$

Combining Eqs. (2.18) to (2.21), we have

$$\begin{aligned}
\|e(t+1)\| &\le \|I_n - \eta(G(t)+H(t))\|\|e(t)\| + \|v(t)\| \\
&\le \Big(\|I_n - \eta G(0)\| + \eta\|G(t)-G(0)\| + \eta\|H(0)\| \\
&\quad + \eta\|H(t)-H(0)\|\Big)\|e(t)\| + \|v(t)\| \\
&\le \Big(1 - \frac{3\eta\gamma}{4} + c_0\frac{\eta n^2\sqrt{\log p}}{\gamma\sqrt{p}} + \frac{\eta\gamma}{4} + c_1\frac{\eta n^2}{\gamma\sqrt{p}} + c_2\frac{\eta n^2\sqrt{\log p}}{\gamma\sqrt{p}} + c_3\frac{\eta^2 n\sqrt{n}}{\sqrt{p}}\Big)\|e(t)\| \\
&\le (1 - \frac{\eta\gamma}{4})\|e(t)\|
\end{aligned}$$

where we use Lemma 2.5.7 and $p = \Omega(\frac{n^4\log p}{\gamma^4})$. □

Proof of Theorem 2.3.2. We prove the inequality (2.4) by induction. Suppose (2.4) and (2.5) hold for all $t = 1, 2, ..., t' - 1$, by Lemma 2.5.8 and Lemma 2.5.9 we know (2.4) and (2.5) hold for $t = t'$, which completes the proof. □

Proof of Theorem 2.4.2

Lemma 2.5.10. *Assume all the inequalities from Lemma 2.5.6 hold. Under the conditions of Theorem 2.4.2, if the error bound (2.7) holds for all $t = 1, 2, ..., t' - 1$, then*

$$\begin{aligned}
\|w_r(t) - w_r(0)\| &\le c_1\frac{n\sqrt{\log p}}{\gamma\sqrt{p}}(1+\eta\tilde{S}_\lambda), \\
|\beta_r(t) - \beta_r(0)| &\le c_2\frac{n}{\gamma\sqrt{p}}(1+\eta\tilde{S}_\lambda)
\end{aligned} \tag{2.22}$$

hold for all $t \le t'$, where c_1, c_2 are constants.

Proof. For any $k \le t' - 1$, we apply (2.7) repeatedly on the right hand side of itself to get

$$\|e(k)\| \leq \prod_{i=0}^{k-1}\Big(1-\frac{\eta\gamma}{4}-\eta\lambda(i)\Big)\|e(0)\| + \sum_{i=0}^{k-1}\eta\lambda(i)\prod_{i<j<k}\Big(1-\frac{\eta\gamma}{4}-\eta\lambda(j)\Big)\|y\|.$$

For $t \leq t'-1$, we take the sum over $k = 0, .., t$ on both sides of above inequality to obtain

$$\begin{aligned}
\sum_{k=0}^{t}\|e(k)\| &\leq \sum_{k=0}^{t}\prod_{i=0}^{k-1}\Big(1-\frac{\eta\gamma}{4}-\eta\lambda(i)\Big)\|e(0)\| + \sum_{k=0}^{t}\sum_{i=0}^{k-1}\eta\lambda(i)\prod_{i<j<k}\Big(1-\frac{\eta\gamma}{4}-\eta\lambda(j)\Big)\|y\| \\
&\leq \sum_{k=0}^{t}\Big(1-\frac{\eta\gamma}{4}\Big)^{k-1}\|e(0)\| + \sum_{k=0}^{t}\sum_{i=0}^{k-1}\eta\lambda(i)\Big(1-\frac{\eta\gamma}{4}\Big)^{k-i-1}\|y\| \\
&\leq \sum_{k=0}^{t}\Big(1-\frac{\eta\gamma}{4}\Big)^{k-1}\|e(0)\| + \eta\|y\|\sum_{k=0}^{t-1}\lambda(i)\sum_{k=i+1}^{T}\Big(1-\frac{\eta\gamma}{4}\Big)^{k-i-1} \\
&\leq \frac{4}{\eta\gamma}\|e(0)\| + \frac{4}{\gamma}\tilde{S}_\lambda\|y\| \\
&\leq \frac{c\sqrt{n}}{\gamma}(\frac{1}{\eta}+\tilde{S}_\lambda)
\end{aligned}$$

where we use $\|e(0)\| = O(\sqrt{n})$ and $\|y\| = O(\sqrt{n})$. Then for all $t \leq t'$, we have

$$\begin{aligned}
|\beta_r(t)-\beta_r(0)| &\leq \frac{\eta}{\sqrt{p}}\sum_{s=0}^{t-1}\sum_{i=1}^{n}|\psi(w_r(t)x_i)e_i(t)| \\
&\leq c\frac{\eta}{\sqrt{p}}\sum_{s=0}^{t-1}\sum_{i=1}^{n}|e_i(t)| \\
&\leq c\frac{\eta\sqrt{n}}{\sqrt{p}}\sum_{s=0}^{t-1}\|e(t)\| \\
&\leq c\frac{\eta\sqrt{n}}{\sqrt{p}}\frac{\sqrt{n}}{\gamma}(\frac{1}{\eta}+\tilde{S}_\lambda) \\
&\leq c\frac{n}{\gamma\sqrt{p}}(1+\eta\tilde{S}_\lambda)
\end{aligned}$$

where we use ψ is bounded and (2.14). We also have

$$\begin{aligned}
\|w_r(t) - w_r(0)\| &\leq \frac{\eta}{\sqrt{p}} \sum_{s=0}^{t-1} \sum_{i=1}^{n} \|\psi'(w_r(t)^\top x_i) b_r x_i e_i(t)\| \\
&\leq c \frac{\eta}{\sqrt{p}} \sum_{s=0}^{t-1} \sum_{i=1}^{n} |b_r||e_i(t)| \\
&\leq c|b_r| \frac{\eta\sqrt{n}}{\sqrt{p}} \sum_{s=0}^{t-1} \|e(t)\| \\
&\leq c|b_r| \frac{\eta\sqrt{n}}{\sqrt{p}} \frac{\sqrt{n}}{\gamma} (\frac{1}{\eta} + \tilde{S}_\lambda) \\
&\leq c \frac{n\sqrt{\log p}}{\gamma\sqrt{p}} (1 + \eta\tilde{S}_\lambda)
\end{aligned}$$

where we use the fact that ψ' is bounded, (2.14) and (2.15). □

Lemma 2.5.11. *Assume all the inequalities from Lemma 2.5.6 hold. Under the conditions of Theorem 2.4.2, if the bound for weights difference (2.22) holds for all $t \leq t'$ and error bound (2.7) holds for all $t \leq t' - 1$, then (2.7) holds for $t = t'$.*

Proof. We start by analyzing the error $e(t)$ according to

$$\begin{aligned}
e_i(t+1) &= \frac{1}{\sqrt{p}} \beta(t+1)^\top \psi(W(t+1)x_i) - y_i \\
&= \frac{1}{\sqrt{p}} \beta(t+1)^\top (\psi(W(t+1)x_i) - \psi(W(t)x_i)) + \frac{1}{\sqrt{p}} (\beta(t+1) - (1-\eta\lambda(t))\beta(t))^\top \psi(W(t)x_{\cdot} \\
&\quad + (1-\eta\lambda(t)) \Big(\frac{1}{\sqrt{p}} \beta(t)^\top \psi(W(t)x_i) - y_i \Big) - \eta\lambda(t) y \\
&= (1-\eta\lambda(t)) e_i(t) - \frac{\eta}{p} \beta(t+1)^\top D_i(t) \sum_{j=1}^{n} D_j(t) b x_j^\top x_i e_j(t) - \frac{\eta}{p} \sum_{j=1}^{n} \psi(W(t)x_j)^\top \psi(W(t)x_i) e_{\cdot} \\
&\quad + v_i(t) \\
&= (1-\eta\lambda(t)) e_i(t) - \eta \sum_{j=1}^{n} \big(H_{ij}(t) + G_{ij}(t) \big) e_j(t) + v_i(t) - \eta\lambda(t) y
\end{aligned}$$

where

$$G_{ij}(t) = \frac{1}{p}\psi(W(t)x_j)^\top \psi(W(t)x_i)$$
$$H_{ij}(t) = \frac{x_i^\top x_j}{p}\beta(t+1)^\top D_i(t)D_j(t)b$$

and $v_i(t)$ is the residual term from a Taylor expansion

$$v_i(t) = \frac{1}{2\sqrt{p}}\sum_{r=1}^{p}\beta_r(t+1)|(w_r(t+1)-w_r(t))^\top x_i|^2\psi''(\xi_{ri}(t))$$

with $\xi_{ri}(t)$ between $w_r(t)^\top x_i$ and $w_r(t+1)^\top x_i$. We can also rewrite the above iteration in vector form as

$$e(t+1) = (1-\lambda(t))e(t) - \eta(G(t)+H(t))e(t) + v(t) - \eta\lambda(t)y. \tag{2.23}$$

Now for $t = t'-1$, we show that both $G(t)$ and $H(t)$ are close to their initialization. Using the argument in Lemma 2.5.9, we can obtain following bounds

$$\|G(t)-G(0)\| \le c_1\frac{n^2\sqrt{\log p}}{\gamma\sqrt{p}}(1+\eta\tilde{S}_\lambda) \tag{2.24}$$

$$\|H(t)-H(0)\| \le c_2\frac{n^2\sqrt{\log p}}{\gamma\sqrt{p}}(1+\eta\tilde{S}_\lambda) \tag{2.25}$$

$$\|v(t)\| \le c_3\frac{\eta^2 n^2}{\sqrt{p}}\|e(t)\|. \tag{2.26}$$

Combining Eqs. (2.23) to (2.26), we have

$$\begin{aligned}
\|e(t+1)\| &\le \|(1-\eta\lambda(t))I_n - \eta(G(t)+H(t))\|\|e(t)\| + \|v(t)\| \\
&\le \Big(\|(1-\eta\lambda(t))I_n - \eta G(0)\| + \eta\|G(t)-G(0)\| + \eta\|H(0)\| \\
&\quad + \eta\|H(t)-H(0)\|\Big)\|e(t)\| + \|v(t)\| \\
&\le \Big(1 - \eta\lambda(t) - \frac{3\eta\gamma}{4} + (c_1+c_2)\frac{\eta n^2\sqrt{\log p}}{\gamma\sqrt{p}}(1+\eta\tilde{S}_\lambda) + c_3\frac{\eta^2 n\sqrt{n}}{\sqrt{p}}\Big)\|e(t)\| \\
&\le (1-\eta\lambda(t) - \frac{\eta\gamma}{4})\|e(t)\|
\end{aligned}$$

where we use Lemma 2.5.7, $p = \Omega(\frac{n^4\log p}{\gamma^4})$ and $\tilde{S}_\lambda = O(\frac{\gamma^2\sqrt{p}}{\eta n^2\sqrt{\log p}})$. □

Proof of Theorem 2.4.2. We prove the inequality (2.7) by induction. Suppose (2.7) holds for all $t = 1, 2, ..., t'-1$. Then by Lemma 2.5.10 and Lemma 2.5.11 we know (2.7) holds for $t = t'$, which completes the proof. □

2.5.3 Alignment on Two-Layer Linear Networks

Now we assume $\psi(u) = u$, so that f is a linear network. The loss function with regularization at time t is

$$\mathcal{L}(t, W, \beta) = \frac{1}{2}\Big\|\frac{1}{\sqrt{p}}XW^\top\beta - y\Big\|^2 + \frac{1}{2}\lambda(t)\|\beta\|^2. \tag{2.27}$$

The regularized feedback alignment algorithm gives

$$\begin{aligned}
W(t+1) &= W(t) - \eta\frac{1}{\sqrt{p}}be(t)^\top X \\
\beta(t+1) &= (1-\eta\lambda(t))\beta(t) - \frac{\eta}{\sqrt{p}}W(t)X^\top e(t)
\end{aligned} \tag{2.28}$$

where $e(t) = \frac{1}{\sqrt{p}}XW(t)^\top\beta(t) - y$ is the error vector at time t.

Lemma 2.5.12. *Suppose the network is trained with the regularized feedback align-*

ment algorithm (2.28)*. Then the prediction error* $e(t)$ *satisfies the recurrence*

$$e(t+1) = \left[(1-\eta\lambda(t))I_d - \frac{\eta}{p}XW(0)^\top W(0)X^\top - \eta\Big(J_1(t)+J_2(t)+J_3(t)\Big)\right]e(t) - \eta\lambda(t)y \tag{2.29}$$

where

$$\begin{aligned}
J_1(t) &= \frac{1}{p}b^\top\beta(0)\prod_{i=0}^{t}(1-\eta\lambda(i))XX^\top \\
J_2(t) &= -\frac{\eta}{p}\Big(\bar{v}^\top X^\top \hat{s}(t)XX^\top + XX^\top s(t-1)\bar{v}^\top X^\top + X\bar{v}s(t-1)^\top XX^\top\Big) \\
J_3(t) &= \frac{\eta^2}{p^2}\|b\|^2\Big(\hat{S}(t)XX^\top + XX^\top s(t-1)s(t-1)^\top XX^\top)
\end{aligned}$$

and

$$\begin{aligned}
\bar{v} &= \frac{1}{\sqrt{p}}W(0)^\top b \\
s(t) &= \sum_{i=0}^{t} e(i) \\
\hat{s}(t) &= \sum_{i=0}^{t}\prod_{i<k\le t}(1-\eta\lambda(k))e(i) \\
\hat{S}(t) &= \sum_{i=0}^{t}\prod_{i<k\le t}(1-\eta\lambda(k))e(i)^\top XX^\top\sum_{j=0}^{i-1}e(j).
\end{aligned}$$

Proof. We first write $W(t)$ in terms of $W(0)$ and $e(i)$, $i \in [t]$, so that

$$W(t) = W(0) - \frac{\eta}{\sqrt{p}}b\sum_{i=0}^{t-1}e(i)^\top X = W(0) - \frac{\eta}{\sqrt{p}}bs(t-1)^\top X. \tag{2.30}$$

Similarly, for $\beta(t)$ we have

$$\begin{aligned}
\beta(t) &= \prod_{i=0}^{t-1}(1-\eta\lambda(i))\beta(0) - \frac{\eta}{\sqrt{p}}\sum_{i=0}^{t-1}\prod_{i<k<t}(1-\eta\lambda(k))W(i)X^\top e(i) \\
&= \prod_{i=0}^{t-1}(1-\eta\lambda(i))\beta(0) - \frac{\eta}{\sqrt{p}}\sum_{i=0}^{t-1}\prod_{i<k<t}(1-\eta\lambda(k))\Big(W(0) - \frac{\eta}{\sqrt{p}}b\sum_{j=0}^{i-1}e(j)^\top X\Big)X^\top e(i) \\
&= \prod_{i=0}^{t-1}(1-\eta\lambda(i))\beta(0) - \frac{\eta}{\sqrt{p}}\sum_{i=0}^{t-1}\prod_{i<k<t}(1-\eta\lambda(k))W(0)X^\top e(i) \\
&\quad + \frac{\eta^2}{p}b\sum_{i=0}^{t-1}\prod_{i<k<t}(1-\eta\lambda(k))e(i)^\top XX^\top\sum_{j=0}^{i-1}e(j) \\
&= \prod_{i=0}^{t-1}(1-\eta\lambda(i))\beta(0) - \frac{\eta}{\sqrt{p}}W(0)X^\top\hat{s}(t-1) + \frac{\eta^2}{p}b\hat{S}(t-1).
\end{aligned} \tag{2.31}$$

We now study how the error $e(t)$ changes after a single update step, writing

$$\begin{aligned}
e(t+1) &= \frac{1}{\sqrt{p}}XW(t+1)^\top\beta(t+1) - y \\
&= \frac{1}{\sqrt{p}}X(W(t+1)-W(t)^\top\beta(t+1) + \frac{1}{\sqrt{p}}XW(t)^\top(\beta(t+1)-(1-\eta\lambda(t))\beta(t)) \\
&\quad + (1-\eta\lambda(t))\Big(\frac{1}{\sqrt{p}}XW(t)^\top\beta(t) - y\Big) - \eta\lambda(t)y \\
&= (1-\eta\lambda(t))e(t) - \frac{\eta}{p}b^\top\beta(t+1)XX^\top e(t) - \frac{\eta}{p}XW(t)^\top W(t)X^\top e(t) - \eta\lambda(t)y
\end{aligned}$$

By plugging (2.30) and (2.31) into above equation, we have

$$\begin{aligned}
e(t+1) = (1-\eta\lambda(t))e(t)& \\
&- \frac{\eta}{p}b^\top\Bigg[\prod_{i=0}^{t}(1-\eta\lambda(i))\beta(0) - \frac{\eta}{\sqrt{p}}W(0)X^\top\hat{s}(t) + \frac{\eta^2}{p}b\hat{S}(t)\Bigg]XX^\top e(t) \\
&- \frac{\eta}{p}X\Bigg[W(0) - \frac{\eta}{\sqrt{p}}bs(t-1)^\top X\Bigg]^\top\Bigg[W(0) - \frac{\eta}{\sqrt{p}}bs(t-1)^\top X\Bigg]X^\top e(t) \\
&- \eta\lambda(t)y
\end{aligned}$$

After expanding the brackets and rearranging the items, we can obtain (2.29). □

Lemma 2.5.13. *Given $\delta \in (0,1)$ and $\epsilon > 0$, if $p = \Omega(\frac{1}{\epsilon}\log\frac{d}{\delta} + \frac{d}{\epsilon}\log\frac{1}{\epsilon})$, the following inequalities hold with probability at least $1-\delta$*

$$\frac{|b^\top \beta(0)|}{\sqrt{p}} \le c\sqrt{\log\frac{1}{\delta}} \tag{2.32}$$

$$\frac{\|b^\top W(0)\|}{\sqrt{p}} \le c\sqrt{d\log\frac{d}{\delta}} \tag{2.33}$$

$$\Big|\frac{\|b\|^2}{p} - 1\Big| \le \frac{c}{\sqrt{p}}\sqrt{\log\frac{1}{\delta}} \tag{2.34}$$

$$\Big\|\frac{1}{p}W(0)^\top W(0) - I_d\Big\| \le \epsilon \tag{2.35}$$

where c is a constant.

Proof. (2.32) is derived from Lemma 2.5.4. (2.33) is by (2.32) and a union bound argument. (2.34) is by Lemma 2.5.3. (2.35) is by Corollary 2.5.2 □

Proof of Theorem 2.4.3. We show (2.8) by induction. Assume (2.8) holds for all $t = 0, 1, ..., t'$, we will show it hold for $t = t'+1$. For any $k \le t'$, we apply (2.8) repeatedly on the right hand side of itself to get

$$\|e(k)\| \le \prod_{i=0}^{k-1}\Big(1 - \frac{\eta\gamma}{2} - \eta\lambda(i)\Big)\|e(0)\| + \sum_{i=0}^{k-1}\eta\lambda(i)\prod_{i<j<k}\Big(1 - \frac{\eta\gamma}{2} - \eta\lambda(j)\Big)\|y\|$$

For $t \leq t'$, we take the sum over $k = 0, .., t$ on both sides of above inequality

$$\begin{aligned}
\sum_{k=0}^{t} \|e(k)\| &\leq \sum_{k=0}^{t} \prod_{i=0}^{k-1} \Big(1 - \frac{\eta\gamma}{2} - \eta\lambda(i)\Big) \|e(0)\| + \sum_{k=0}^{t} \sum_{i=0}^{k-1} \eta\lambda(i) \prod_{i<j<k} \Big(1 - \frac{\eta\gamma}{2} - \eta\lambda(j)\Big) \|y\| \\
&\leq \sum_{k=0}^{t} \Big(1 - \frac{\eta\gamma}{2}\Big)^{k-1} \|e(0)\| + \sum_{k=0}^{t} \sum_{i=0}^{k-1} \eta\lambda(i) \Big(1 - \frac{\eta\gamma}{2}\Big)^{k-i-1} \|y\| \\
&\leq \sum_{k=0}^{t} \Big(1 - \frac{\eta\gamma}{2}\Big)^{k-1} \|e(0)\| + \eta\|y\| \sum_{k=0}^{t-1} \lambda(i) \sum_{k=i+1}^{T} \Big(1 - \frac{\eta\gamma}{2}\Big)^{k-i-1} \\
&\leq \frac{2}{\eta\gamma}\|e(0)\| + \frac{2}{\gamma} S_\lambda \|y\| \\
&\leq \frac{c\sqrt{n}}{\gamma}(\frac{1}{\eta} + S_\lambda)
\end{aligned}$$

where we use $\|e(0)\| = O(\sqrt{n})$ and $\|y\| = O(\sqrt{n})$. With this bound and the inequalities from Lemma 2.5.13, we can bound the norms of $J_1(t)$, $J_2(t)$ and $J_3(t)$ from Lemma 2.5.12. It follows that

$$\|J_1(t)\| \leq \frac{1}{p}|b^\top \beta(0)|\|XX^\top\| \leq c\frac{M\sqrt{\log \delta^{-1}}}{\sqrt{p}} \leq \frac{\gamma}{16}, \tag{2.36}$$

$$\|J_2(t)\| \leq \frac{\eta}{p}\|X\|\|XX^\top\|\|\bar{v}\|(2\|s(t-1)\| + \|\hat{s}(t)\|) \leq c\frac{\eta}{p}M^{3/2}\sqrt{d \log \frac{d}{\delta}}\frac{\sqrt{n}}{\gamma}(\frac{1}{\eta} + S_\lambda) \leq \frac{\gamma}{16} \tag{2.37}$$

and

$$\|J_3(t)\| \leq \frac{\eta^2}{p^2}\|b\|^2(\|XX^\top\||\hat{S}(t)| + \|XX^\top\|^2\|s(t-1)\|^2) \leq c\frac{\eta^2}{p}M^2\frac{n}{\gamma^2}(\frac{1}{\eta} + S_\lambda)^2 \leq \frac{\gamma}{16} \tag{2.38}$$

hold for all $t \leq t'$ if $p = \Omega(\frac{Md\log(d/\delta)}{\gamma})$ and $S_\lambda = O(\frac{\gamma\sqrt{\gamma p}}{\eta\sqrt{n}M})$. Furthermore, since

$\|\frac{1}{p}W(0)W(0)^\top - I_d\| \le \epsilon_0$ with high probability when $p = \Omega(d)$, we have

$$\begin{aligned}\|\frac{1}{p}XW(0)^\top W(0)X^\top - \gamma I_d\| &\le \|\frac{1}{p}XW(0)^\top W(0)X^\top - XX^\top\| + \|XX^\top - \gamma I_d\| \\ &\le (1+\epsilon)\epsilon_0\gamma + \epsilon\gamma \le \frac{\gamma}{16}\end{aligned} \tag{2.39}$$

Therefore, combining (2.36), (2.37), (2.38) and (2.29), we have

$$\begin{aligned}\|e(t'+1)\| &\le \Big(1 - \eta\lambda(t') - \eta\gamma\Big)\|e(t')\| + \eta\Big\|\frac{\eta}{p}XW(0)^\top W(0)X^\top - \gamma I_d\Big\|\|e(t')\| \\ &\quad + \eta(\|J_1(t')\| + \|J_2(t')\| + \|J_3(t')\|)\|e(t')\| + \eta\lambda(t')\|y\| \\ &\le \Big(1 - \eta\lambda(t') - \eta\gamma\Big)\|e(t')\| + \frac{1}{16}\eta\gamma\|e(t')\| + \frac{3}{16}\eta\gamma\|e(t')\| + \eta\lambda(t')\|y\| \\ &\le \Big(1 - \eta\lambda(t') - \frac{\eta\gamma}{2}\Big)\|e(t')\| + \eta\lambda(t')\|y\|\end{aligned}$$

which completes the proof. □

Proof of Proposition 2.4.5. By Corollary 2.5.2, if $d = \Omega(\frac{1}{\epsilon}\log\frac{n}{\delta} + \frac{n}{\epsilon}\log\frac{1}{\epsilon})$, we have

$$\|XX^\top - I_n\| \le \epsilon$$

It follows that $\lambda_{\min}(XX^\top) \ge 1-\epsilon$ and $\lambda_{\max}(XX^\top) \le 1+\epsilon \le (1+4\epsilon)(1-\epsilon)$ for $\epsilon < 1/2$. □

Lemma 2.5.14. *Recall from Lemma 2.5.12 that*

$$\beta(t) = \prod_{i=0}^{t-1}(1-\eta\lambda(i))\beta(0) - \frac{\eta}{\sqrt{p}}W(0)X^\top\hat{s}(t-1) + \frac{\eta^2}{p}b\hat{S}(t-1)$$

with $\hat{s}(t) = \sum_{i=0}^{t}\prod_{i<k\le t}(1-\eta\lambda(k))e(i)$ *and* $\hat{S}(t) = \sum_{i=0}^{t}\prod_{i<k\le t}(1-\eta\lambda(k))e(i)^\top XX^\top\sum_{j=0}^{i-1}e(j)$. *Under the conditions of Theorem 2.4.6, if* $t > C_1\frac{\log(p/\eta)}{\eta\lambda}$ *and* $\hat{S}(t) \ge \max(C_2\frac{\sqrt{p\gamma}}{\eta}\|\hat{s}(t)\|, 1)$ *for some positive constants* C_1 *and* C_2*, then* $\cos\angle(b, \beta(t)) \ge c$ *for some constant* $c = c_\delta$.

Proof. We compute the cosine of the angle between $\beta(t)$ and b. With probability $1-\delta$,

$$\begin{aligned}\cos\angle(b,\beta(t)) &= \frac{b^\top\beta(t)}{\|b\|\|\beta(t)\|} = \frac{\frac{b}{\|b\|}^\top\beta(t)}{\|\beta(t)\|}\\ &\geq \frac{\frac{\eta^2}{p}\|b\|\hat{S}(t-1) - (1-\eta\lambda)^t\|\beta(0)\| - \frac{\eta}{\sqrt{p}}\|\frac{b}{\|b\|}^\top W(0)\|\|X\|\|\hat{s}(t-1)\|}{\frac{\eta^2}{p}\|b\|\hat{S}(t-1) + (1-\eta\lambda)^t\|\beta(0)\| + \frac{\eta}{\sqrt{p}}\|W(0)\|\|X\|\|\hat{s}(t-1)\|}\\ &\geq \frac{c_1'\frac{\eta^2}{\sqrt{p}}\hat{S}(t-1) - c_2'\sqrt{p}(1-\eta\lambda)^t - c_3'\eta\sqrt{\frac{d\gamma}{p}}\|\hat{s}(t-1)\|}{c_1'\frac{\eta^2}{\sqrt{p}}\hat{S}(t-1) + c_2'\sqrt{p}(1-\eta\lambda)^t + c_4'\eta\sqrt{\gamma}\|\hat{s}(t-1)\|}\end{aligned}$$

where we use (2.34), (2.35) and the tail bound for standard Gaussian vectors, and c_i' are constants that only depend on δ. Notice that if $t=\Omega(\frac{\log(p/\eta)}{\eta\lambda})$, we have $c_2'\sqrt{p}(1-\eta\lambda)^t = O(\frac{\eta^2}{\sqrt{p}})$. It follows that $\cos\angle(b,\beta(t))\geq c$ if $\hat{S}(t-1)=\Omega(\frac{\sqrt{p\gamma}}{\eta}\|\hat{s}(t-1)\|+1)$. $\square$

Lemma 2.5.15. *Consider the orthogonal decomposition $e(t)=a(t)\bar{y}+\xi(t)$, where $\bar{y}=-y/\|y\|$ and $\xi(t)\perp y$. Under the conditions of Theorem 2.4.6, there exists a constant $C_\tau>0$ such that for any $t\in[\tau,T]$ with $\tau=\frac{C_\tau}{\eta\lambda}$, we have*

$$a(t)\geq\frac{\lambda-\gamma}{\lambda+\gamma}\|y\| \tag{2.40}$$

and

$$\|\xi(t)\|\leq\frac{\gamma}{\lambda+\gamma}\|y\|. \tag{2.41}$$

Proof. By Theorem 2.4.3, we have for all $t\leq T$, $\|e(t)\|\leq(1-\eta\lambda-\eta\gamma/2)\|e(t)\|+\eta\lambda\|y\|$. By rearranging the terms, we have

$$\|e(t+1)\|-\frac{\lambda}{\lambda-\gamma/2}\|y\|\leq(1-\eta\lambda-\frac{\eta\gamma}{2})\Big(\|e(t)\|-\frac{\lambda}{\lambda-\gamma/2}\|y\|\Big)$$

or

$$\|e(t)\|-\frac{\lambda}{\lambda-\gamma/2}\|y\|\leq(1-\eta\lambda-\frac{\eta\gamma}{2})^t\Big(\|e_0\|-\frac{\lambda}{\lambda-\gamma/2}\|y\|\Big)\leq(1-\eta\lambda)^t(\|e_0\|+\|y\|).$$

Notice that $\|y\|$ and $\|e(0)\|$ are of the same order, so when $t \in [\tau_1, T]$ with $\tau_1 = \frac{c_1}{\eta\lambda}$ and some constant c_1, we have

$$\|e(t)\| \leq \frac{\lambda + \gamma/2}{\lambda - \gamma/2}\|y\|. \tag{2.42}$$

In order to get a lower bound for $a(t)$, we multiply $\bar{y}^\top$ on both sides of (2.29). It follows that for $t \in [\tau_1, T]$

$$\begin{aligned} a(t+1) \geq \bar{y}^\top \Big(1 - \eta\lambda - \eta\gamma\Big)e(t) - \eta\|\frac{1}{p}XW(0)^\top W(0)X^\top - \gamma I_d\|\|e(t)\| \\ - \eta(\|J_1(t)\| + \|J_2(t)\| + \|J_3(t)\|)\|e(t)\| + \eta\lambda\|y\| \\ \geq (1 - \eta\lambda - \eta\gamma)a(t) - \frac{1}{4}\eta\gamma\|e(t)\| + \eta\lambda\|y\| \\ \geq (1 - \eta\lambda - \eta\gamma)a(t) + \frac{1}{2}\eta\gamma\|y\|. \end{aligned}$$

In the second inequality, we use the bounds (2.36), (2.37), (2.38) and (2.39). The last inequality is by (2.42) and $\lambda \geq 3\gamma$. Following a similar derivation, we have

$$a(t) - \frac{\lambda - \gamma/2}{\lambda + \gamma}\|y\| \geq (1 - \eta\lambda - \eta\gamma)^{t-\tau_1}\Big(a(\tau_1) - \frac{\lambda - \gamma/2}{\lambda + \gamma}\|y\|\Big) \geq -(1 - \eta\lambda)^{t-\tau_1}(\|e(\tau_1)\| + \|y\|).$$

The bound (2.40) holds when $t \in [\tau_1 + \tau_2, T]$ with $\tau_2 = \frac{c_2}{\eta\lambda}$ and some constant c_2. Then we multiply $\frac{\xi(t+1)^\top}{\|\xi(t+1)\|}$ on both sides of (2.29). This establishes that for $t \in [\tau_1, T]$

$$\begin{aligned} \|\xi(t+1)\| \leq \frac{\xi(t+1)^\top}{\|\xi(t+1)\|}\Big(1 - \eta\lambda - \eta\gamma\Big)e(t) + \eta\|\frac{1}{p}XW(0)^\top W(0)X^\top - \gamma I_d\|\|e(t)\| \\ + \eta(\|J_1(t)\| + \|J_2(t)\| + \|J_3(t)\|)\|e(t)\| + \eta\lambda\|y\| \\ \leq (1 - \eta\lambda - \eta\gamma)\|\xi(t)\| + \frac{\eta\gamma}{4}\|e(t)\| \\ \leq (1 - \eta\lambda - \eta\gamma)\|\xi(t)\| + \frac{\eta\gamma}{2}\eta\gamma\|y\|. \end{aligned}$$

The first inequality is by $\xi(t+1)^\top y = 0$ and in the second inequality we use $\xi(t +$

$1)^\top e(t) = \xi(t+1)^\top \xi(t) \le \|\xi(t+1)\|\|\xi(t)\|$. It follows that

$$\|\xi(t)\| - \frac{\gamma/2}{\lambda+\gamma}\|y\| \le (1-\eta\lambda-\eta\gamma)^{t-\tau_1}\Big(\|\xi(0)\| - \frac{\gamma/2}{\lambda+\gamma}\|y\|\Big) \le (1-\eta\lambda)^{t-\tau_1}(\|e(\tau_1)\|+\|y\|).$$

The bound (2.41) holds when $t \in [\tau_1+\tau_3, T]$ with $\tau_3 = \frac{c_3}{\eta\lambda}$ for a constant c_3. Finally, the bounds (2.40) and (2.41) hold when $t \in [\tau, T]$ with $\tau = \tau_1 + \max(\tau_2, \tau_3)$. $\square$

Lemma 2.5.16. *Under the conditions of Theorem 2.4.6, suppose* $T = \lfloor \frac{S_\lambda}{\lambda} \rfloor = C_T \frac{\sqrt{p}}{\eta\sqrt{n\gamma}}$*. Then we have* $\hat{S}(T) \ge \tilde{c}\frac{\sqrt{p\gamma}}{\eta}\|\hat{s}(T)\|$*, where* C_T *and* $\tilde{c}$ *are positive constants.*

Proof. Notice that

$$e(i)^\top XX^\top e(j) \ge \gamma e(i)^\top e(j) - \|e(i)\|\|e(j)\|\|XX^\top - \gamma I\| \ge \gamma e(i)^\top e(j) - \epsilon\gamma\|e(i)\|\|e(j)\|.$$

For $i \in [T/2, T]$ and τ defined in Lemma 2.5.15, we have

$$\begin{aligned} e(i)^\top XX^\top \sum_{j<i} e(j) &= e(i)^\top XX^\top \sum_{\tau \le j < i} e(j) + e(i)^\top XX^\top \sum_{j<\tau} e(j) \\ &\ge \sum_{\tau\le j<i}\Big(\gamma e(i)^\top e(j) - \epsilon\gamma\|e(i)\|\|e(j)\|\Big) - 2\gamma\sum_{j<\tau}\|e(i)\|\|e(j)\| \\ &\ge \sum_{\tau\le j<i}\gamma\Big(a(i)a(j) - \|\xi(i)\|\|\xi(j)\| - \epsilon\|e(i)\|\|e(j)\|\Big) - 2c\tau\gamma\|y\|^2 \\ &\ge (i-\tau)\gamma\Big[\Big(\frac{\lambda-\gamma}{\lambda+\gamma}\Big)^2\|y\|^2 - \Big(\frac{\gamma}{\lambda+\gamma}\Big)^2\|y\|^2 - \epsilon\Big(\frac{\lambda+\gamma/2}{\lambda-\gamma/2}\Big)^2\|y\|^2 - \frac{2c\tau}{i-\tau}\|y\|^2\Big] \\ &\ge \frac{T}{8}\gamma\|y\|^2 = \frac{C_T}{8}\frac{\sqrt{p}}{\eta\sqrt{n\gamma}}\gamma\|y\|^2 \\ &\ge c\frac{\sqrt{p\gamma}}{\eta}\|y\|. \end{aligned} \tag{2.43}$$

The second inequality is the orthogonal decomposition of $e(i)$ and $\|e(i)\| \le c\|y\|$ given by (2.8). The third inequality is by (2.40), (2.41) and (2.42) from Lemma 2.5.15. The fourth inequality is by $\lambda = \Omega(\gamma)$, $i - \tau \ge T/4$ and the fact that $\tau/(i-\tau)$ is small

$(p = \Omega(n))$. The last inequality is by $\|y\| = \Theta(\sqrt{n})$. Therefore,

$$\begin{aligned}
\hat{S}(T) &= \sum_{i=0}^{T}(1-\eta\lambda)^{T-i}e(i)^\top XX^\top \sum_{j<i} e(j) \\
&= \sum_{i=T/2}^{T}(1-\eta\lambda)^{T-i}e(i)^\top XX^\top \sum_{j<i} e(j) + (1-\eta\lambda)^{T/2}\sum_{i=0}^{T/2}(1-\eta\lambda)^{T/2-i}e(i)^\top XX^\top \sum_{j<i} e(j) \\
&\geq \sum_{i=T/2}^{T}(1-\eta\lambda)^{T-i}c\frac{\sqrt{p\gamma}}{\eta}\|y\| + (1-\eta\lambda)^{T/2}\sum_{i=0}^{T/2}(1-\eta\lambda)^{T/2-i}c'T\gamma\|y\|^2 \\
&\geq \frac{c}{2}\frac{\sqrt{p\gamma}}{\eta}\frac{\|y\|}{\eta\lambda} - (1-\eta\lambda)^{T/2}\frac{c'T\gamma\|y\|^2}{\eta\lambda} \\
&\geq \frac{c}{4}\frac{\sqrt{p\gamma}}{\eta}\frac{\|y\|}{\eta\lambda}
\end{aligned}$$

where the last inequality is by $(1-\eta\lambda)^{T/2} \ll 1$ when $p = \Omega(n)$. On the other hand,

$$\|\hat{s}(T)\| \leq \sum_{i=0}^{T}(1-\eta\lambda)^{T-i}\|e(i)\| \leq \frac{c}{\eta\lambda}\|y\|.$$

Combining the above inequalities gives the proof. □

Proof of Theorem 2.4.6. First, notice that $\lambda(t) = 0$ when $t > T$. By Theorem 2.4.3 we have that the prediction error converges to zero exponentially fast, or $\|e(t+1)\| \leq (1-\eta\gamma/2)\|e(t)\|$. It follows that $\hat{S}(t) \to \hat{S}(\infty)$ and $\hat{s}(t) \to \hat{s}(\infty)$ as $t \to \infty$. By Lemma 2.5.14, we know it suffices to show $\hat{S}(\infty) \geq C\frac{\sqrt{p\gamma}}{\eta}\|\hat{s}(\infty)\|$ with some constant C. Since

$$\hat{S}(\infty) = \sum_{i=0}^{\infty}(1-\eta\lambda)^{(T-i)_+}e(i)^\top XX^\top \sum_{j<i} e(j) = \hat{S}(T) + \sum_{i>T} e(i)^\top XX^\top \sum_{j<i} e(j)$$

and

$$\hat{s}(\infty) = \sum_{i=0}^{\infty}(1-\eta\lambda)^{(T-i)_+}e(i) = \hat{s}(T) + \sum_{i>T} e(i),$$

by Lemma 2.5.16, it suffices to show

$$\sum_{i>T} e(i)^\top XX^\top \sum_{j<i} e(j) \geq C\frac{\sqrt{p\gamma}}{\eta}\sum_{i>T}\|e(i)\|. \tag{2.44}$$

We write $g = XX^\top \sum_{j<T} e(j)$. Then we have

$$\begin{aligned}
\|g\| &\geq \lambda_{\min}(XX^\top)\Big[\Big\|\sum_{\tau\geq j<T} e(j)\Big\| - \sum_{j<\tau}\|e(j)\|\Big] \\
&\geq \lambda_{\min}(XX^\top)\Big[\sum_{\tau\geq j<T} a(j) - \sum_{j<\tau}\|e(j)\|\Big] \\
&\geq \gamma\Big[(T-\tau)\Big(\frac{\lambda-\gamma}{\lambda+\gamma}\Big)\|y\| - \tau c\|y\|\Big]
\end{aligned} \tag{2.45}$$

and

$$\begin{aligned}
\|g\| &\leq \|XX^\top\|\Big(\sum_{j<\tau}\|e(j)\| + \sum_{\tau\geq j<T}\|e(j)\|\Big) \\
&\leq (1+\epsilon)\gamma\Big[\tau c\|y\| + (T-\tau)\Big(\frac{\lambda+\gamma/2}{\lambda-\gamma/2}\Big)\|y\|\Big]
\end{aligned} \tag{2.46}$$

where we use the bounds (2.40) and (2.42) from Lemma 2.5.15. We further denote $\alpha(t) = \bar{g}^\top e(t)$ where $\bar{g} = g/\|g\|$. Following the same calculation in (2.43), we have

$$\begin{aligned}
g^\top e(T) &= e(T)^\top XX^\top \sum_{j<T} e(j) \\
&\geq (T-\tau)\gamma\Big[\Big(\frac{\lambda-\gamma}{\lambda+\gamma}\Big)^2\|y\|^2 - \Big(\frac{\gamma}{\lambda+\gamma}\Big)^2\|y\|^2 - \epsilon\Big(\frac{\lambda+\gamma/2}{\lambda-\gamma/2}\Big)^2\|y\|^2 - \frac{2c\tau}{T-\tau}\|y\|^2\Big].
\end{aligned}$$

Then

$$\begin{aligned}
\frac{\alpha(T)}{\|e(T)\|} &\geq \frac{g^\top e(T)}{\|g\|\|e(T)\|} \\
&\geq \frac{(T-\tau)\gamma\Big[\Big(\frac{\lambda-\gamma}{\lambda+\gamma}\Big)^2\|y\|^2 - \Big(\frac{\gamma}{\lambda+\gamma}\Big)^2\|y\|^2 - \epsilon\Big(\frac{\lambda+\gamma/2}{\lambda-\gamma/2}\Big)^2\|y\|^2 - \frac{2c\tau}{T-\tau}\|y\|^2\Big]}{(1+\epsilon)\gamma\Big[\tau c\|y\| + (T-\tau)\Big(\frac{\lambda+\gamma/2}{\lambda-\gamma/2}\Big)\|y\|\Big] \times \Big(\frac{\lambda+\gamma/2}{\lambda-\gamma/2}\Big)\|y\|} \\
&\geq \frac{\Big[\Big(\frac{\lambda-\gamma}{\lambda+\gamma}\Big)^2 - \Big(\frac{\gamma}{\lambda+\gamma}\Big)^2 - \epsilon\Big(\frac{\lambda+\gamma/2}{\lambda-\gamma/2}\Big)^2 - \frac{2c\tau}{T-\tau}\Big]}{(1+\epsilon)\Big[\frac{\tau c}{T-\tau} + \Big(\frac{\lambda+\gamma/2}{\lambda-\gamma/2}\Big)\Big] \times \Big(\frac{\lambda+\gamma/2}{\lambda-\gamma/2}\Big)}.
\end{aligned}$$

Notice that $T/\tau = \Omega(\sqrt{p/n})$, so that when p/n, λ/γ are large and ϵ is small, we have

$$\alpha(T) \geq \frac{3}{4}\|e(T)\|. \tag{2.47}$$

In order to obtain the lower bound on $\alpha(t)$ for all $t \geq T$, we multiply $\bar{g}^\top$ on both sides of (2.29). Notice $\lambda(t) = 0$ and apply the bounds (2.36), (2.37), (2.38) and (2.39). We have that

$$\begin{aligned}\alpha(t+1) &\geq (1-\eta\gamma)\bar{g}^\top e(t) - \eta\|\frac{1}{p}XW(0)^\top W(0)X^\top - \gamma I_d\|\|e(t)\| \\ &\quad - \eta(\|J_1(t)\| + \|J_2(t)\| + \|J_3(t)\|)\|e(t)\| \\ &\geq (1-\eta\gamma)\alpha(t) - \frac{\eta\gamma}{4}\|e(t)\|\end{aligned}$$

or for $t \geq T$,

$$\alpha(t) \geq (1-\eta\gamma)^{t-T}\alpha(T) - \frac{\eta\gamma}{4}\sum_{i=T}^{t-1}(1-\eta\gamma)^{t-i}\|e(i)\|. \tag{2.48}$$

Taking the sum over $t > T$, we have

$$\begin{aligned}\sum_{t>T}\alpha(t) &\geq \sum_{t>T}(1-\eta\gamma)^{t-T}\alpha(T) - \frac{\eta\gamma}{4}\sum_{t>T}\sum_{i=T}^{t-1}(1-\eta\gamma)^{t-i}\|e(i)\| \\ &\geq \frac{1-\eta\gamma}{\eta\gamma}\alpha(T) - \frac{\eta\gamma}{4}\sum_{i>T}\|e(i)\|\sum_{t>i}(1-\eta\gamma)^{t-i} \\ &\geq \frac{1-\eta\gamma}{\eta\gamma}\Big(\alpha(T) - \frac{\eta\gamma}{4}\sum_{i>T}\|e(i)\|\Big) \\ &\geq \frac{1-\eta\gamma}{\eta\gamma}(\alpha(T) - \frac{1}{2}\|e(T)\|) \\ &\geq \frac{1-\eta\gamma}{4\eta\gamma}\|e(T)\|.\end{aligned} \tag{2.49}$$

The second inequality follows from switching the order of sums. The fourth inequality is by exponential convergence after T steps. The last inequality is by (2.47). With the above inequalities, we are ready to bound the left hand side of (2.44), obtaining

$$\begin{aligned}
\sum_{i>T} e(i)^\top XX^\top \sum_{j<i} e(j) &= \sum_{i>T} e(i)^\top XX^\top \sum_{j<T} e(j) + \sum_{i>T} e(i)^\top XX^\top \sum_{j\geq T} e(j)\\
&\geq \sum_{t>T} \alpha(t)\|g\| - 2\gamma\Big(\sum_{i\geq t}\|e(i)\|\Big)^2\\
&\geq \frac{1-\eta\gamma}{4\eta\gamma}\|e(T)\|\gamma\Big[(T-\tau)\Big(\frac{\lambda-\gamma}{\lambda+\gamma}\Big)\|y\| - \tau c\|y\|\Big] - 2\gamma\frac{4}{\eta^2\gamma^2}\|e(T)\|^2\\
&\geq \frac{1-\eta\gamma}{4\eta\gamma}\|e(T)\|\gamma\Big[(T-\tau)\Big(\frac{\lambda-\gamma}{\lambda+\gamma}\Big)\|y\| - \tau c\|y\| - \frac{64}{\eta\gamma(1-\eta\gamma)}\|y\|\Big]\\
&\geq \frac{1-\eta\gamma}{4\eta\gamma}\|e(T)\|\gamma\frac{T}{2}\|y\| = \frac{1-\eta\gamma}{4\eta\gamma}\|e(T)\|\gamma\frac{C_T}{2}\frac{\sqrt{p}}{\eta\sqrt{n\gamma}}\|y\|\\
&\geq C\frac{1-\eta\gamma}{4\eta\gamma}\frac{\sqrt{p\gamma}}{\eta}\|e(T)\|.
\end{aligned} \tag{2.50}$$

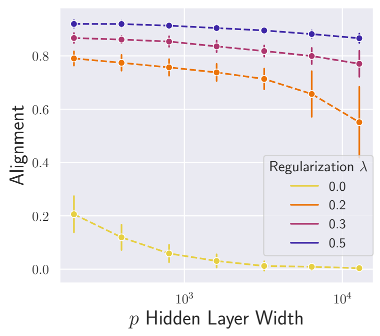
The second inequality is by (2.49) and (2.45). The third inequality is by $\|e(T)\| \leq 2\|y\|$. The last inequality is by $\|y\| = \Theta(\sqrt{n})$. On the other hand,

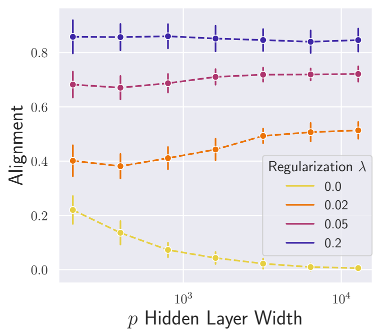
$$\sum_{i>T}\|e(i)\| \leq \sum_{i>T}(1-\eta\gamma/2)^{i-T}\|e(T)\| = \frac{1-\eta\gamma/2}{\eta\gamma/2}\|e(T)\| \tag{2.51}$$

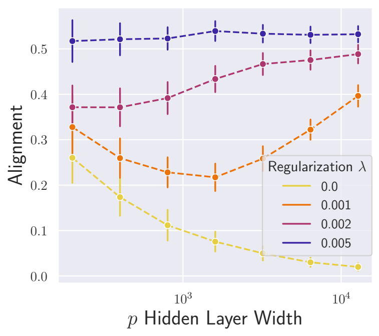
Combining (2.50) and (2.51) implies (2.44), as desired. □

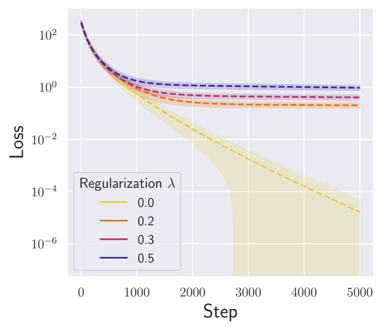
2.6 Simulations

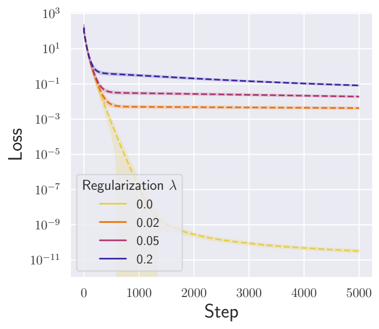
Our experiments apply the feedback alignment algorithm to two-layer networks, using a range of networks with different widths and activations. The numerical results suggest that regularization is essential in achieving alignment, in both regression and classification tasks, for linear and nonlinear models. We implement the feedback alignment procedure in PyTorch as an extension of the autograd module for backpropagation, and the training is done on V100 GPUs from internal clusters.


(a) Alignment on linear network.


(b) Alignment on ReLU network.


(c) Alignment on Tanh network.


(d) Loss on linear network.


(e) Loss on ReLU network.

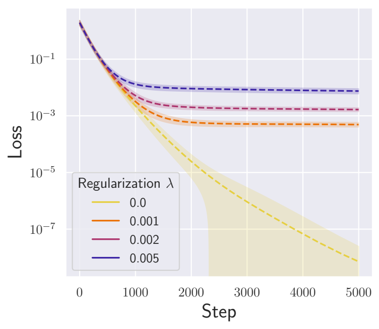

(f) Loss on Tanh network.

Figure 2.2: Comparisons of alignment and convergence for the feedback alignment algorithm on synthetic data with different levels of ℓ_2 regularization. The x-axes on the first row and the y-axes on the second row are presented using a logarithmic scale.

Feedback alignment on synthetic data. We first train two-layer networks on synthetic data, where each network f shares the architecture shown in (2.1) and the data are generated by another network f_0 that has the same architecture but with random Gaussian weights. We present the experiments for both linear and nonlinear networks, where the activation functions are chosen to be Rectified Linear Unit (ReLU) and hyperbolic tangent (Tanh) for nonlinear case. We set training sample sample size to $n = 50$ and the input dimension $d = 150$, but vary the hidden layer width $p = 100 \times 2^k$ with $k \in [7]$. During training, we take step size $\eta = 10^{-4}$ for linear networks and $\eta = 10^{-3}, 10^{-2}$ for ReLU and Tanh networks, respectively.

In Figs. 2.2a to 2.2c, we show how alignment depends on regularization and the degree of overparameterization as measured by the hidden layer width p. Alignment is measured by the cosine of the angle between the forward weights β and backward weights b. We train the networks until the loss function converges; this procedure is repeated 50 times for each p and λ. The data points in the figures represent the mean value computed across simulations, and the error bars mark the standard deviation out of 50 independent runs. For all three types of networks, as p increases, alignment vanishes if there is no regularization, and grows with the level of regularization λ for the same network. We complement the alignment plots with the corresponding loss curves. In Figs. 2.2d to 2.2f, we show the trajectories of the training loss for networks with $p = 3200$, with the shaded areas indicating the standard deviation over 50 independent runs. We observe that the training loss converges slower with larger regularization. These numerical results are consistent with our theoretical statements. Due to the regularization, the loss converges to a positive number that is of the same order as λ.

We remark that using dropout as a form of regularization can also help the alignment between forward and backward weights (Wager et al., 2013). However, our numerical results suggest that dropout regularization fails to keep the alignment away from zero for networks with large hidden layer width. No theoretical result is available that explains the underlying mechanism.

Feedback alignment on the MNIST dataset. The `MNIST` dataset is available under the Creative Commons Attribution-Share Alike 3.0 license (Deng, 2012). It consists of 60,000 training images and 10,000 test images of dimension 28 by 28. We reshape them into vectors of length $d = 784$ and normalize them by their mean and standard deviation. The network structure is 784-1000-10 with ReLU activation at the hidden layer and with softmax normalization at output layer. During training, we choose the batch size to be 600 and the step size $\eta = 10^{-2}$. The training procedure uses 300 epochs in total. We repeat the training 10 times for each choice of λ.

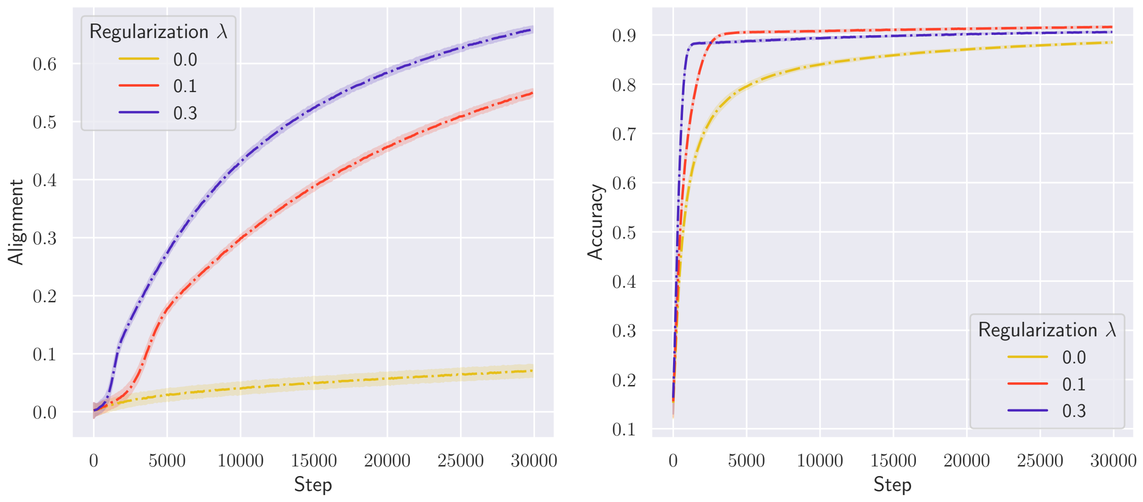


Figure 2.3: Comparisons of alignment and accuracy for feedback alignment algorithm on `MNIST` with different levels of ℓ_2 regularization.

Fig. 2.3 shows the performance of feedback alignment with regularization $\lambda = 0, 0.1, 0.3$. The dashed lines and corresponding shaded areas represent the means and the standard deviations over 10 runs with random initialization. Since the output of the network is not one-dimensional but 10-dimensional, the alignment is now measured by $\cos\angle(\delta_{\rm BP}(h), \delta_{\rm FA}(h))$, where $\delta_{\rm BP}(h)$ is the error signal propagated to the hidden neurons h through forward weights β, and $\delta_{\rm FA}(h)$ the error weighted by the random backward weights b. We observe that both alignment and convergence are improved by adding regularization to the training, and increasing the regularization level λ can further facilitate alignment, with a small gain in test accuracy.

2.7 Discussion

In this chapter we have analyzed the feedback alignment algorithm of Lillicrap et al. (2016), showing convergence of the algorithm. The convergence is subtle, as the algorithm does not directly minimize the target loss function; rather, the error is transferred to the hidden neurons through random weights that do not change during the course of learning. The supplement to Lillicrap et al. (2016) presents interesting insights on the dynamics of the algorithm, such as how the feedback weights act as pseudoinverse of the forward weights. After giving an analysis of convergence in the linear case, the authors state that "a general proof must be radically different from those used to demonstrate convergence for backprop" (Supplementary note 16), observing that the algorithm does not minimize any loss function. Our proof of convergence in the general nonlinear case leverages techniques from the use of neural tangent kernel analysis in the over-parameterized setting, but requires more care because the kernel is not positive semi-definite at initialization. In particular, as a sum of two terms G and H, the matrix G is concentrated around its positive-definite mean, while H is not generally positive-semidefinite. However, we show that the entries of both matrices remain close to their initial values, due to over-parameterization, and analyze the error term in a Taylor expansion, which establishes convergence.

In analyzing alignment, we unexpectedly found that regularization is essential; without it, the alignment may not persist as the network becomes wider, as our simulations clearly show. Our analysis in the linear case proceeds by essentially showing that

$$\beta(t) = (1-\eta\lambda)^{t-1}\beta(0) + \frac{\eta}{\sqrt{p}}W(0)X^\top\alpha_1(t-1) + \left(\frac{\eta}{\sqrt{p}}\right)b\alpha_2(t-1)$$

and controlling α_1 while showing that α_2 remains sufficiently large; the regularization kills off the first term. Although we see no obstacle, in principle, to carrying out this proof strategy in the nonlinear case, the calculations are more complex. While convergence requires analysis of the norm of the error, alignment requires understanding the direction of the error. But our simulations strongly suggest this result will go through.

2.8 Future Work

In this work, we provide convergence results on general two-layer networks and alignment results on linear networks, without an activation function. An immediate future direction is to analyze the alignment phenomenon on two-layer nonlinear networks, and it is also meaningful to study the training problem in the large sample regime rather than the over-parameterization regime. In the multi-layer case, direct feedback alignment (DFA) (Nøkland, 2016) provides an alternative training scheme where the error signals are propagated directly to each layer with a single random weight matrix, instead of being propagated layer by layer. The theoretical analysis for both FA and DFA on multi-layer networks would be important in the future.

In terms of methodology, it would be interesting to explore new algorithms to improve the performance of the feedback alignment algorithm. Specifically, in feedback alignment, the backward weights between layers are fixed, but it seems more natural to update them during training. One possible approach is to use Hebbian learning rule to devise a local update method for the backward weights. Another way could be updating the backward weights by the input signals propagated through forward weights, which can be viewed as an analogy of updating the forward weights by the error signals propagated through backward weights in feedback alignment. We would expect that both the convergence and alignment can be improved when updating weights in both directions.

More generally, it is also important to study other biologically plausible learning rules that can be implemented in deep learning frameworks at scale and without loss of performance. The results presented here offer support for this as a fruitful line of research.

Chapter 3

Surfing: Iterative Optimization Over Incrementally Trained Deep Networks

In this chapter, we investigate a sequential optimization procedure to minimize the empirical risk functional $f_{\hat{\theta}}(x) = \frac{1}{2}\|G_{\hat{\theta}}(x) - y\|^2$ for certain families of deep networks $G_\theta(x)$. The approach is to optimize a sequence of objective functions that use network parameters obtained during different stages of the training process. When initialized with random parameters θ_0, we show that the objective $f_{\theta_0}(x)$ is “nice” and easy to optimize with gradient descent. As learning is carried out, we obtain a sequence of generative networks $x \mapsto G_{\theta_t}(x)$ and associated risk functions $f_{\theta_t}(x)$, where t indicates a stage of stochastic gradient descent during training. Since the parameters of the network do not change by very much in each step, the surface evolves slowly and can be incrementally optimized. The algorithm is formalized and analyzed for a family of expansive networks. We call the procedure *surfing* since it rides along the peak of the evolving (negative) empirical risk function, starting from a smooth surface at the beginning of learning and ending with a wavy nonconvex surface after learning is complete. Experiments show how surfing can be used to find the global optimum and for compressed sensing even when direct gradient descent on the final learned network fails.

3.1 Introduction

Intensive recent research has provided insight into the performance and mathematical properties of deep neural networks, improving understanding of their strong empirical performance on different types of data. Some of this work has investigated gradient descent algorithms that optimize the weights of deep networks during learning (Du et al., 2018b,a; Davis et al., 2018; Li and Yuan, 2017; Li and Liang, 2018). In this chapter we focus on optimization over the inputs to an already trained deep network in order to best approximate a target data point. Specifically, we consider the least squares objective function

$$f_{\hat{\theta}}(x) = \frac{1}{2}\|G_{\hat{\theta}}(x) - y\|^2$$

where $G_\theta(x)$ denotes a multi-layer feed-forward network and $\hat{\theta}$ denotes the parameters of the network after training. The network is considered to be a mapping from a latent input $x \in \mathbb{R}^k$ to an output $G_\theta(x) \in \mathbb{R}^n$ with $k \ll n$. A closely related objective is to minimize $f_{\theta,A}(x) = \frac{1}{2}\|AG_\theta(x) - Ay\|^2$ where A is a random matrix.

Hand and Voroninski (2019) study the behavior of the function $f_{\theta_0,A}$ in a compressed sensing framework where $y = G_{\theta_0}(x_0)$ is generated from a random network with parameters $\theta_0 = (W_1, \ldots, W_d)$ drawn from Gaussian matrix ensembles; thus, the network is not trained. In this setting, it is shown that the surface is very well behaved. In particular, outside of small neighborhoods around x_0 and a scalar multiple of $-x_0$, the function $f_{\theta_0,A}(x)$ always has a descent direction.

When the parameters of the network are trained, the landscape of the function

$f_{\hat{\theta}}(x)$ can be complicated; it will in general be nonconvex with multiple local optima. Figure 1 illustrates the behavior of the surfaces as they evolve from random networks (left) to fully trained networks (right) for 4-layer networks trained on Fashion MNIST using a variational autoencoder. For each of two target values y, three surfaces $x \mapsto -\frac{1}{2}\|G_{\theta_t}(x) - y\|^2$ are shown for different levels of training.

This chapter explores the following simple idea. We incrementally optimize a sequence of objective functions $f_{\theta_0}, f_{\theta_1}, \ldots, f_{\theta_T}$ where the parameters $\theta_0, \theta_1, \ldots, \theta_T = \hat{\theta}$ are obtained using stochastic gradient descent in θ during training. When initialized with random parameters θ_0, we show that the empirical risk function $f_{\theta_0}(x) = \frac{1}{2}\|G_{\theta_0}(x) - y\|^2$ is "nice" and easy to optimize with gradient descent. As learning is carried out, we obtain a sequence of generative networks $x \mapsto G_{\theta_t}(x)$ and associated risk functions $f_{\theta_t}(x)$, where t indicates an intermediate stage of stochastic gradient descent during training. Since the parameters of the network do not change by very much in each step (Du et al., 2018a,b), the surface evolves slowly. We initialize x for the current network $G_{\theta_t}(x)$ at the optimum x^*_{t-1} found for the previous network $G_{\theta_{t-1}}(x)$ and then carry out gradient descent to obtain the updated point $x^*_t = \operatorname{argmin}_x f_{\theta_t}(x)$.

We call this process *surfing* since it rides along the peaks of the evolving (negative) empirical risk function, starting from a smooth surface at the beginning of learning and ending with a wavy nonconvex surface after learning is complete. We formalize this algorithm in a manner that makes it amenable to analysis. First, when θ_0 is initialized so that the weights are random Gaussian matrices, we prove a theorem showing that the surface has a descent direction at each point outside of a small neighborhood. The analysis of Hand and Voroninski (2019) does not directly apply in our case since the target y is an arbitrary test point, and not necessarily generated according to the random network. We then give an analysis that describes how projected gradient descent can be used to proceed from the optimum of one network to the next. Our approach is based on the fact that the ReLU network and squared

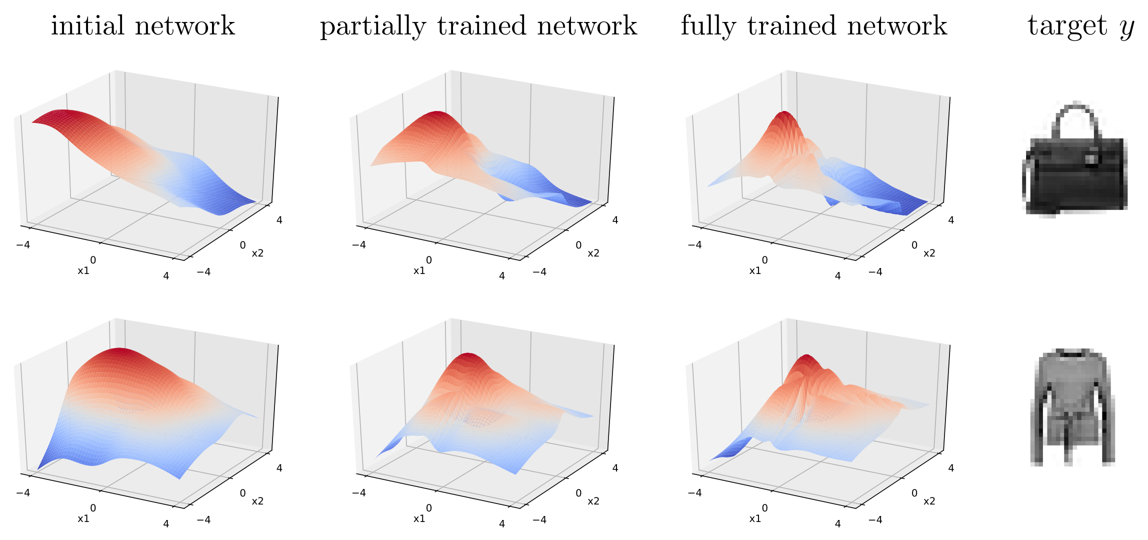


Figure 3.1: Behavior of the surfaces $x \mapsto -\frac{1}{2}\|G_{\theta_t}(x) - y\|^2$ for two targets y shown for three levels of training, from random networks (left) to fully trained networks (right) on Fashion MNIST data.

error objective result in a piecewise quadratic surface. Experiments are run to show how surfing can be used to find the global optimum and for compressed sensing even when direct gradient descent fails, using several experimental setups with networks trained with both VAE and GAN techniques.

3.2 Background and Previous Results

In this work we treat the problem of approximating an observed vector y in terms of the output $G_{\hat{\theta}}(x)$ of a trained generative model. Traditional generative processes such as graphical models are statistical models that define a distribution over a sample space. When deep networks are viewed as generative models, the distribution is typically singular, being a deterministic mapping of a low-dimensional latent random vector to a high-dimensional output space. Certain forms of "reversible deep networks" allow for the computation of densities and inversion (Dinh et al., 2017; Kingma and Dhariwal, 2018; Chen et al., 2018).

The variational autoencoder (VAE) approach training a generative (decoder) network is to model the conditional probability of x given y as Gaussian with mean $\mu(y)$ and covariance $\Sigma(y)$ assuming that *a priori* $x \sim N(0, I_k)$ is Gaussian. The mean and covariance are treated as the output of a secondary (encoder) neural network. The two networks are trained by maximizing the evidence lower bound (ELBO) with coupled gradient descent algorithms—one for the encoder network, the other for the decoder network $G_\theta(x)$ (Kingma and Welling, 2014). Whether fitting the networks using a variational or GAN approach (Goodfellow et al., 2014; Arjovsky et al., 2017), the problem of "inverting" the network to obtain $x^* = \operatorname{argmin} f_\theta(x)$ is not addressed by the training procedure.

In the now classical compressed sensing framework (Candes et al., 2006; Donoho et al., 2006), the problem is to reconstruct a sparse signal after observing multiple linear measurements, possibly with added noise. More recent work has begun to investigate generative deep networks as a replacement for sparsity in compressed sensing. Bora et al. (2017) consider identifying $y = G(x_0)$ from linear measurements Ay by optimizing $f(x) = \frac{1}{2}\|Ay - AG(x)\|^2$. Since this objective is nonconvex, it is not guaranteed that gradient descent will converge to the true global minimum. However, for certain classes of ReLU networks it is shown that so long as a point $\hat{x}$ is found for which $f(\hat{x})$ is sufficiently close to zero, then $\|y - G(\hat{x})\|$ is also small. For the case where y does not lie in the image of G, an oracle type bound is shown implying that the solution $\hat{x}$ satisfies $\|G(\hat{x}) - y\|^2 \leq C \inf_x \|G(x) - y\|^2 + \delta$ for some small error term δ. The authors observe that in experiments the error seems to converge to zero when $\hat{x}$ is computed using simple gradient descent; but an analysis of this phenomenon is not provided.

Hand and Voroninski (2019) establish the important result that for a d-layer random network and random measurement matrix A, the least squares objective has favorable geometry, meaning that outside two small neighborhoods there are no first order stationary points, neither local minima nor saddle points. We describe their setup and result in some detail, since it provides a springboard for the surfing algorithm. Let $G : \mathbb{R}^k \to \mathbb{R}^n$ be a d-layer fully connected feedforward generative neural network, which has the form $G(x) = \sigma(W_d ... \sigma(W_2 \sigma(W_1 x))...)$ where σ is the ReLU activation function. The matrix $W_i \in R^{n_i \times n_{i-1}}$ is the set of weights for the ith layer and n_i is number of the neurons in this layer with $k = n_0 < n_1 < ... < n_d = n$. If $x_0 \in \mathbb{R}^k$ is the input then $AG(x_0)$ is a set of random linear measurements of the signal $y = G(x_0)$. The objective is to minimize $f_{A,\theta_0}(x) = \frac{1}{2}\left\| AG_{\theta_0}(x) - AG_{\theta_0}(x_0) \right\|^2$ where $\theta_0 = (W_1, \ldots, W_d)$ is the set of weights.

Due to the fact that the nonlinearities σ are rectified linear units, $G_{\theta_0}(x)$ is a piecewise linear function. It is convenient to introduce notation that absorbs the activation σ into weight matrix W_i, denoting

$$W_{+,x} = \operatorname{diag}(Wx > 0)W.$$

For a fixed W, the matrix $W_{+,x}$ zeros out the rows of W that do not have a positive dot product with x; thus, $\sigma(Wx) = W_{+,x}x$. We further define $W_{1,+,x} = \operatorname{diag}(W_1 x > 0)\, W_1$ and

$$W_{i,+,x} = \operatorname{diag}(W_i W_{i-1,+,x} ... W_{1,+,x} x > 0)\, W_i.$$

With this notation, we can rewrite the generative network G_{θ_0} in what looks like a linear form,

$$G_{\theta_0}(x) = W_{d,+,x} W_{d-1,+,x} ... W_{1,+,x} x,$$

noting that each matrix $W_{i,+,x}$ depends on the input x. If $f_{A,\theta_0}(x)$ is differentiable at x, we can write the gradient as

$$\nabla f_{A,\theta_0}(x) = \Big(\prod_{i=d}^{1} W_{i,+,x}\Big)^T A^T A \Big(\prod_{i=d}^{1} W_{i,+,x}\Big) x - \Big(\prod_{i=d}^{1} W_{i,+,x}\Big)^T A^T A \Big(\prod_{i=d}^{1} W_{i,+,x_0}\Big) x_0.$$

In this expression, one can see intuitively that under the assumption that A and W_i are Gaussian matrices, the gradient $\nabla f_{\theta_0}(x)$ should concentrate around a deterministic vector v_{x,x_0}. Hand and Voroninski (2019) establish sufficient conditions for concentration of the random matrices around deterministic quantities, so that v_{x,x_0} has norm bounded away from zero if x is sufficiently far from x_0 or a scalar multiple of $-x_0$. Their results show that for random networks having a sufficiently expansive number of neurons in each layer, the objective f_{A,θ_0} has a landscape favorable to gradient descent.

We build on these ideas, showing first that optimizing with respect to x for a random network and arbitrary signal y can be done with gradient descent. This requires modified proof techniques, since it is no longer assumed that $y = G_{\theta_0}(x_0)$. In fact, y can be arbitrary and we wish to approximate it as $G_{\hat{\theta}}(x(y))$ for some $x(y)$. Second, after this initial optimization is carried out, we show how projected gradient descent can be used to track the optimum as the network undergoes a series of small changes. Our results are stated formally in the following section.

3.3 Theoretical Results

Suppose we have a sequence of networks $G_0, G_1, \ldots, G_T$ generated from the training process. For instance, we may take a network with randomly initialized weights as G_0, and record the network after each step of gradient descent in training; $G_T = G$ is the final trained network.

For a given vector $y \in \mathbb{R}^n$, we wish to minimize the objective $f(x) = \frac{1}{2}\|AG(x) - Ay\|^2$ with respect to x for the final network G, where either $A = I \in \mathbb{R}^{n\times n}$, or $A \in \mathbb{R}^{m\times n}$ is a measurement matrix with i.i.d. $\mathbb{N}(0, 1/m)$ entries in a compressed

Algorithm 3 Surfing

```
Input: Sequence of networks $\theta_0, \theta_1, \dots, \theta_T$
1: $x_{-1} \leftarrow 0$
2: for $t = 0$ to $T$ do
3:     $x \leftarrow x_{t-1}$
4:     repeat
5:         $x \leftarrow x - \eta \nabla f_{\theta_t}(x)$
6:     until convergence
7:     $x_t \leftarrow x$
8: end for
Output: $x_T$
```

sensing context. Write

$$f_t(x) = \frac{1}{2}\|AG_t(x) - Ay\|^2, \quad \forall\, t \in [T]. \tag{3.1}$$

The idea is that we first minimize f_0, which has a nicer landscape, to obtain the minimizer x_0. We then apply gradient descent on f_t for $t = 1, 2, ..., T$ successively, starting from the minimizer x_{t-1} for the previous network.

We provide some theoretical analysis in partial support of this algorithmic idea. First, we show that at random initialization G_0, all critical points of $f_0(x)$ are localized to a small ball around zero. Second, we show that if $G_0, \dots, G_T$ are obtained from a discretization of a continuous flow, along which the global minimizer of $f_t(x)$ is unique and Lipschitz-continuous, then a projected-gradient version of surfing can successively find the minimizers for $G_1, \dots, G_T$ starting from the minimizer for G_0.

We consider expansive feedforward neural networks $G : \mathbb{R}^k \times \Theta \mapsto \mathbb{R}^n$ given by

$$G(x, \theta) = V\sigma(W_d \dots \sigma(W_2\sigma(W_1 x + b_1) + b_2) \dots + b_d).$$

Here, d is the number of intermediate layers (which we will treat as constant), σ is the ReLU activation function $\sigma(x) = \max(x, 0)$ applied entrywise, and $\theta = (V, W_1, ..., W_d, b_1, ..., b_d)$ are the network parameters. The input dimension is $k \equiv n_0$, each intermediate layer $i \in [d]$ has weights $W_i \in \mathbb{R}^{n_i \times n_{i-1}}$ and biases $b_i \in \mathbb{R}^{n_i}$, and a linear transform $V \in \mathbb{R}^{n \times n_d}$ is applied in the final layer.

For our first result, consider fixed $y \in \mathbb{R}^n$ and a random initialization $G_0(x) \equiv G(x, \theta_0)$ where θ_0 has Gaussian entries (independent of y). If the network is sufficiently expansive at each intermediate layer, then the following shows that with high probability, all critical points of $f_0(x)$ belong to a small ball around 0. More concretely, the directional derivative $D_{-x/\|x\|} f_0(x)$ satisfies

$$D_{-x/\|x\|} f_0(x) \equiv \lim_{t \to 0^+} \frac{f_0(x - tx/\|x\|) - f_0(x)}{t} < 0. \tag{3.2}$$

Thus $-x/\|x\|$ is a first-order descent direction of the objective f_0 at x.

Theorem 3.3.1. *Fix $y \in \mathbb{R}^n$. Let V have $\mathcal{N}(0, 1/n)$ entries, let b_i and W_i have $\mathcal{N}(0, 1/n_i)$ entries for each $i \in [d]$, and suppose these are independent. There exist d-dependent constants $C, C', c, \varepsilon_0 > 0$ such that for any $\varepsilon \in (0, \varepsilon_0)$, if*

1. *$n \geq n_d$ and $n_i > C(\varepsilon^{-2} \log \varepsilon^{-1}) n_{i-1} \log n_i$ for all $i \in [d]$, and*

2. *Either $A = I$ and $m = n$, or $A \in \mathbb{R}^{m \times n}$ has i.i.d. $\mathrm{N}(0, 1/m)$ entries (independent of $V, \{b_i\}, \{W_i\}$) where $m \geq Ck(\varepsilon^{-1} \log \varepsilon^{-1}) \log(n_1 \ldots n_d)$,*

then with probability at least $1 - C(e^{-c\varepsilon m} + n_d e^{-c\varepsilon^4 n_{d-1}} + \sum_{i=1}^{d-1} n_i e^{-c\varepsilon^2 n_{i-1}})$, every $x \in \mathbb{R}^k$ outside the ball $\|x\| \leq C'\varepsilon(1 + \|y\|)$ satisfies (3.2).

We defer the proof to Section 3.4. Note that if instead G_0 were correlated with y, say $y = G_0(x_*)$ for some input x_* with $\|x_*\| \asymp 1$, then x_* would be a global minimizer of $f_0(x)$, and we would have $\|y\| \asymp \|x_d\| \asymp \ldots \asymp \|x_1\| \asymp \|x_*\| \asymp 1$ in the above network where $x_i \in \mathbb{R}^{n_i}$ is the output of the ith layer. The theorem shows that for a random initialization of G_0 which is independent of y, the minimizer is instead localized to a ball around 0 which is smaller in radius by the factor ε.

For our second result, consider a network flow

$$G^s(x) \equiv G(x, \theta(s))$$

for $s \in [0, S]$, where $\theta(s) = (V(s), W_1(s), b_1(s), \ldots, W_d(s), b_d(s))$ evolve continuously in a time parameter s. As a model for network training, we assume that $G_0, \ldots, G_T$ are obtained by discrete sampling from this flow via $G_t = G^{\delta t}$, corresponding to $s \equiv \delta t$ for a small time discretization step δ.

We assume boundedness of the weights and uniqueness and Lipschitz-continuity of the global minimizer along this flow.

Assumption 3.3.2. *There are constants $M, L < \infty$ such that*

1. *For every $i \in [d]$ and $s \in [0, S]$,*

$$\|W_i(s)\| \leq M.$$

2. *The global minimizer $x_*(s) = \operatorname{argmin}_x f(x, \theta(s))$ is unique and satisfies*

$$\|x_*(s) - x_*(s')\| \leq L|s - s'|$$

where $f(x, \theta(s)) = \frac{1}{2}\|AG(x, \theta(s)) - Ay\|^2$.

Fixing θ, the function $G(x, \theta)$ is continuous and piecewise-linear in x. For each $x \in \mathbb{R}^k$, there is at least one linear piece P_0 (a polytope in $\mathbb{R}^k$) of this function that contains x. For a slack parameter $\tau > 0$, consider the rows given by

$$S(x, \theta, \tau) = \{(i, j) : |w_{i,j}^\top x_{i-1} + b_{i,j}| \leq \tau\},$$

where

$$x_{i-1} = \sigma(W_{i-1} \ldots \sigma(W_1 x + b_1) \ldots + b_{i-1})$$

is the output of the $(i-1)^{\text{th}}$ layer for this input x, and $v_j^\top$, $w_{i,j}^\top$, and $b_{i,j}$ are respectively the j^{th} row of V, the j^{th} row of W_i and the j^{th} entry of b_i in θ. This set $S(x, \theta, \tau)$ represents those neurons that are close to 0 before ReLU thresholding, and hence whose activations may change after a small change of the network input x. Define

$$\mathcal{P}(x, \theta, \tau) = \{P_0, P_1, \ldots, P_G\}$$

as the set of all linear pieces P_g whose activation patterns differ from P_0 only in rows belonging to $S(x, \theta, \tau)$. That is, for every $x' \in P_g \in \mathcal{P}(x, \theta, \tau)$ and $(i, j) \notin S(x, \theta, \tau)$, we have

$$\text{sign}(w_{i,j}^\top x'_{i-1} + b_{i,j}) = \text{sign}(w_{i,j}^\top x_{i-1} + b_{i,j})$$

where x'_{i-1} is the output of the $(i-1)^{\text{th}}$ layer for input x'.

With this definition, we consider a stylized projected-gradient surfing procedure in Algorithm 4, where Proj_P is the orthogonal projection onto the polytope P.

Algorithm 4 Projected-gradient Surfing

```
Input: Network flow $\{G(\cdot, \theta(s)) : s \in [0, S]\}$, parameters $\delta, \tau, \eta > 0$.
 1: Initialize $x_0 = \text{argmin}_x f(x, \theta(0))$.
 2: for $t = 1, \ldots, T$ do
 3:     for each linear piece $P_g \in \mathcal{P}(x_{t-1}, \theta(\delta t), \tau)$ do
 4:         $x \leftarrow x_{t-1}$
 5:         repeat
 6:             $x \leftarrow \text{Proj}_{P_g}(x - \eta \nabla f(x, \theta(\delta t)))$
 7:         until convergence
 8:         $x_t^{(g)} \leftarrow x$
 9:     end for
10:     $x_t \leftarrow x_t^{(g)}$ for $g \in \{0, \ldots, G\}$ that achieves the minimum value of $f(x_t^{(g)}, \theta(\delta t))$.
11: end for
Output: $x_T$
```

The complexity of this algorithm depends on the number of pieces G to be optimized over in each step. We expect this to be small in practice when the slack parameter τ is chosen sufficiently small. In fact, the projected-gradient surfing algorithm performs an exhaustive search over pieces $P_g \in \mathcal{P}(x_{t-1}, \theta(\delta t), \tau)$. The number of such pieces is at most $1 + 2^{|S(x_{t-1}, \theta(\delta t), \tau)|}$, where we recall that

$$S(x, \theta, \tau) = \{(i, j) : |w_{i,j}^\top x_{i-1} + b_{i,j}| \leq \tau\}$$

is the collection of layers and rows where the sign could change during the next step. We reason heuristically that if $\theta \equiv \theta(\delta t)$ is "generic", then for sufficiently small τ, we should have $|S(x, \theta, \tau)| \leq dk$ for all $s \in [0, S]$ and $x \in \mathbb{R}^k$, so that this search is tractable for small k. Indeed, for fixed $W_1, b_1, \ldots, W_i, b_i$, the set of possible outputs $\{x_i : x \in \mathbb{R}^k\}$ at the i^{th} layer is a finite union of affine linear spaces of dimension k. For generic W_{i+1} and b_{i+1}, and every $J \subset [n_i]$ where $|J| = k + 1$, each such space has empty intersection with the affine linear space

$$\{z \in \mathbb{R}^{n_i} : w_{i+1,j}^\top z + b_{i+1,j} = 0 \text{ for all } j \in J\}$$

of dimension $n_i - k - 1$. Thus

$$\sup_{x \in \mathbb{R}^k} |\{j \in [n_i] : w_{i+1,j}^\top x_i + b_{i+1,j} = 0\}| \leq k,$$

so $\sup_{x \in \mathbb{R}^k} |S(x, \theta, 0)| \leq dk$ for $\tau = 0$. Then we expect this to hold also for some small $\tau > 0$.

The following shows that for any $\tau > 0$, there is a sufficiently fine time discretization δ depending on τ, M, L such that Algorithm 4 tracks the global minimizer. In particular, for the final objective $f_T(x) = f(x, \theta(\delta T))$ corresponding to the network G_T, the output x_T is the global minimizer of $f_T(x)$. We remark that the time discretization δ may need to be smaller for deeper networks, as $G(x)$ corresponding to a deeper network may have a larger Lipschitz constant in x. The specific dependence below arises from bounding this Lipschitz constant by $\prod_{i=1}^d \|W_i\|$, which is a conservative bound also used and discussed in greater detail in Szegedy et al. (2014); Virmaux and Scaman (2018).

Theorem 3.3.3. *Suppose Assumption 3.3.2 holds. For any $\tau > 0$, if $\delta < \tau/(L\max(M,1)^{d+1})$ and $x_0 = \operatorname{argmin}_x f(x, \theta(0))$, then the iterates x_t in Algorithm 4 are given by $x_t = \operatorname{argmin}_x f(x, \theta(\delta t))$ for each $t = 1, \ldots, T$.*

Proof. For any fixed θ, let $x, x' \in \mathbb{R}^k$ be two inputs to $G(x, \theta)$. If x_i, x_i' are the corresponding outputs of the i^{th} layer, using the assumption $\|W_i\| \leq M$ and the fact that the ReLU activation σ is 1-Lipschitz, we have

$$\begin{aligned} \|x_i - x_i'\| &= \|\sigma(W_i x_{i-1} + b_i) - \sigma(W_i x_{i-1}' + b_i)\| \\ &\leq \|(W_i x_{i-1} + b_i) - (W_i x_{i-1}' + b_i)\| \\ &\leq M\|x_{i-1} - x_{i-1}'\| \leq \ldots \leq M^i \|x - x'\|. \end{aligned}$$

Let $x_*(s) = \operatorname{argmin}_x f(x, \theta(s))$. By assumption, $\|x_*(s-\delta) - x_*(s)\| \leq L\delta$. For the network with parameter $\theta(s)$ at time s, let $x_{*,i}(s)$ and $x_{*,i}(s-\delta)$ be the outputs at the i^{th} layer corresponding to inputs $x_*(s)$ and $x_*(s-\delta)$. Then for any $i \in [d]$ and $j \in [n_i]$, the above yields

$$\begin{aligned} |(w_{i,j}(s)^\top x_{*,i}(s-\delta) + b_{i,j}) - (w_{i,j}(s)^\top x_{*,i}(s) + b_{i,j})| &\leq \|w_{i,j}(s)\| \|x_{*,i}(s-\delta) - x_{*,i}(s)\| \\ &\leq M \cdot M^i \|x_*(s-\delta) - x_*(s)\| \leq M^{i+1} L\delta. \end{aligned}$$

For $\delta < \tau/(L\max(M,1)^{d+1})$, this implies that for every (i,j) where $|w_{i,j}(s)^\top x_{*,i}(s-\delta) + b_{i,j}| \geq \tau$, we have

$$\operatorname{sign}(w_{i,j}(s)^\top x_{*,i}(s-\delta) + b_{i,j}) = \operatorname{sign}(w_{i,j}(s)^\top x_{*,i}(s) + b_{i,j}).$$

That is, $x_*(s) \in P_g$ for some $P_g \in \mathcal{P}(x_*(s-\delta), \theta(s), \tau)$.

Assuming that $x_{t-1} = x_*(\delta(t-1))$, this implies that the next global minimizer $x_*(\delta t)$ belongs to some $P_g \in \mathcal{P}(x_{t-1}, \theta(\delta t), \tau)$. Since $f(x, \theta(\delta t))$ is quadratic on P_g, projected gradient descent over P_g in Algorithm 4 converges to $x_*(\delta t)$, and hence Algorithm 4 yields $x_t = x_*(\delta t)$. The result then follows from induction on t. $\square$

3.4 Proof of Theorem 3.3.1

We denote $[n] = \{1, 2, ..., n\}$, $\Pi_{i=1}^{d} W_i = W_1 W_2 \ldots W_d$, and $\Pi_{i=d}^{1} W_i = W_d W_{d-1} \cdots W_1$. $\|x\|$ and $\|A\|$ are the Euclidean vector norm and matrix operator norm. $C, C', c, c' > 0$ denote d-dependent constants that may change from instance to instance.

We adapt ideas of Hand and Voroninski (2019). Denote for simplicity $G(x) = G(x, \theta_0)$ and $f(x) = f_0(x)$. Define

$$W_{i,+,v} = \mathrm{diag}(W_i v + b_i > 0) W_i, \qquad b_{i,+,v} = \mathrm{diag}(W_i v + b_i > 0) b_i$$

where $\mathrm{diag}(w > 0)$ denotes a diagonal matrix with jth diagonal element $\mathbb{I}\{w_j > 0\}$. Then

$$\sigma(W_i v + b_i) = W_{i,+,v} v + b_{i,+,v}.$$

The analysis of Hand and Voroninski (2019) shows that the matrices

$$\tilde{W}_{i,+,v} \equiv \begin{pmatrix} W_{i,+,v} & b_{i,+,v} \end{pmatrix} \in \mathbb{R}^{n_i \times (n_{i-1}+1)}$$

satisfy a certain Weight Distribution Condition (WDC), yielding a deterministic approximation for $\tilde{W}_{i,+,v}^\top \tilde{W}_{i,+,v'}$ and any $v, v' \in \mathbb{R}^{n_{i-1}}$. We will use the following consequence of this condition.

Lemma 3.4.1. *Under the conditions of Theorem 3.3.1, with probability at least* $1 - C\sum_{i=1}^{d} n_i e^{-c\varepsilon^2 n_{i-1}}$*, the following hold for every* $i \in [d]$ *and* $v, v' \in \mathbb{R}^{n_{i-1}}$*:*

(a) $\|W_{i,+,v}\| \le 2$ *and* $\|b_{i,+,v}\| \le 2$.

(b) $\|W_{i,+,v}^\top W_{i,+,v'} - \frac{1}{2}I\| \le \varepsilon + \theta/\pi$, *where* θ *is the angle formed by* v *and* v'.

(c) $\|W_{i,+,v}^\top b_{i,+,v}\| \le \varepsilon$.

Proof. For (a), note that $\|W_i\| \le 2$ and $\|b_i\| \le 2$ with probability $1 - e^{-cn_i}$, by a standard χ^2 tail-bound and operator norm bound for a Gaussian matrix. On the event that these hold, the bounds hold also for $W_{i,+,v}$ and $b_{i,+,v}$ and every $v \in \mathbb{R}^{n_{i-1}}$.

For (b) and (c), by (Hand and Voroninski, 2019, Lemma 11), with probability $1 - 8n_i e^{-c\varepsilon^2 n_{i-1}}$ the matrix $\tilde{W}_{i,+,v}$ satisfies WDC with constant ε for every v. (The dependence of the constants c, γ in (Hand and Voroninski, 2019, Lemma 11) are given by $c \gtrsim \varepsilon^{-2} \log \varepsilon^{-1}$ and $\gamma \lesssim \varepsilon^2$ as indicated in the proof. This condition for c matches the growth rate of n_i specified in our Theorem 3.3.1.) From the form of Q in (Hand and Voroninski, 2019, Definition 2), the WDC implies

$$\left\| \tilde{W}_{i,+,v}^\top \tilde{W}_{i,+,v'} - \frac{1}{2}I \right\| \le \varepsilon + \tilde{\theta}/\pi$$

where $\tilde{\theta}$ is the angle between $(v, 1)$ and $(v', 1)$. Noting that $\tilde{\theta} \le \theta$ and recalling the definition of $\tilde{W}_{i,+,v}$, we get (b) and (c). □

For $x \in \mathbb{R}^k$, let $x_0 = x$ and let $x_i = \sigma(W_i \ldots \sigma(W_1 x + b_1) \ldots + b_i)$ be the output of the ith layer. Denote

$$W_{i,x} = W_{i,+,x_{i-1}}, \qquad b_{i,x} = b_{i,+,x_{i-1}}.$$

Then also $x_i = W_{i,x} x_{i-1} + b_{i,x}$.

Lemma 3.4.2. *Under the conditions of Theorem 3.3.1, with probability 1, the total number of distinct possible tuples* $(W_{1,x}, b_{1,x}, \ldots, W_{d,x}, b_{d,x})$ *satisfies*

$$|\{(W_{1,x}, b_{1,x}, \ldots, W_{d,x}, b_{d,x}) : x \in \mathbb{R}^k\}| \le 10^{d^2} (n_1 \ldots n_d)^{d(k+1)}.$$

Proof. Let $S = \mathbb{R}^{k+1}$, which contains $(x, 1)$. Then the result of (Hand and Voroninski, 2019, Lemma 15) applied to the vector space S and to $\tilde{W}_{1,x} = (W_{1,x} \;\; b_{1,x})$ yields

$$|\{(W_{1,x}, b_{1,x} : x \in \mathbb{R}^k)\}| \leq 10n_1^{k+1}.$$

Each distinct $(W_{1,x}, b_{1,x})$ defines an affine linear space of dimension k which contains the first layer output x_1, and hence a subspace S of dimension $k+1$ which contains $(x_1, 1)$. Applying (Hand and Voroninski, 2019, Lemma 15) to each such S and $\tilde{W}_{2,x}$ yields

$$|\{(W_{2,x}, b_{2,x} : x \in \mathbb{R}^k)\}| \leq 10n_1^{k+1} \cdot 10n_2^{k+1}.$$

Proceeding inductively,

$$|\{(W_{i,x}, b_{i,x} : x \in \mathbb{R}^k)\}| \leq 10^i (n_1 \dots n_i)^{k+1},$$

which is analogous to (Hand and Voroninski, 2019, Lemma 16) in our setting with biases $b_1, \dots, b_d$. The result follows from taking the product over $i = 1, \dots, d$. □

Lemma 3.4.3. *Let $A \in \mathbb{R}^{m \times n}$ have i.i.d. $\mathbb{N}(0, 1/m)$ entries. Fix $\varepsilon > 0$, let $k < n$, and let $V = \bigcup_{i=1}^M V_i$ and $W = \bigcup_{j=1}^N W_j$ where V_i and W_j are subspaces of dimension at most k. Then with probability at least $1 - MN(c/\varepsilon)^{2k} e^{-c'\varepsilon m}$, for all $x \in V$ and $y \in W$ we have*

$$|x^\top A^\top A y - x^\top y| \leq \varepsilon \|x\| \|y\|.$$

Proof. See (Hand and Voroninski, 2019, Lemma 14). □

Using these results, we analyze the gradient and critical points of $f(x)$. Note that with the above definitions,

$$\begin{aligned}G(x) &= V(W_{d,x}\dots(W_{1,x}x + b_{1,x})\dots + b_{d,x})\\ &= V\left(\prod_{i=d}^{1} W_{i,x}\right)x + V\sum_{j=1}^{d}\left(\prod_{i=d}^{j+1} W_{i,x}\right)b_{j,x}.\end{aligned}$$

The function $G(x)$ is piecewise linear in x, so $f(x)$ is piecewise quadratic. If $f(x)$ is differentiable at x, then the gradient of f can be written as

$$\nabla f(x) = \left(\prod_{i=1}^{d} W_{i,x}^{\top}\right)V^{\top}A^{\top}\left(AV\left(\prod_{i=d}^{1} W_{i,x}\right)x + AV\sum_{j=1}^{d}\left(\prod_{i=d}^{j+1} W_{i,x}\right)b_{j,x} - Ay\right).$$

Lemma 3.4.4. *Define*

$$g_x = 2^{-d}x - \left(\prod_{i=1}^{d} W_{i,x}^{\top}\right)V^{\top}y$$

Under the conditions of Theorem 3.3.1, we have with probability $1 - C(e^{-c\varepsilon m} + e^{-c\varepsilon n} + \sum_i n_i e^{-c\varepsilon^2 n_{i-1}})$ *that at every* $x \in \mathbb{R}^k$ *where* f *is differentiable,*

$$\|\nabla f(x) - g_x\| \le C'\varepsilon(1 + \|x\| + \|y\|)$$

Proof. By Lemma 3.4.2, for fixed $\theta = (V, W_1, b_1, \dots, W_d, b_d)$, the range $\{V\prod_{i=d}^{1} W_{i,x}x' : x, x' \in \mathbb{R}^k\}$ belongs to a union of at most $C(n_1\dots n_d)^{d(k+1)}$ subspaces of dimension k. For some $C', c > 0$, under the condition $m \ge C'k(\varepsilon^{-1}\log\varepsilon^{-1})\log(n_1\dots n_d)$, we have

$$C^2(n_1\dots n_d)^{2d(k+1)}(c/\varepsilon)^{2k}e^{-c'\varepsilon m} \le e^{-c\varepsilon m}.$$

Then for $A \in \mathbb{R}^{m\times n}$ with i.i.d. $\mathbb{N}(0, 1/m)$ entries, applying Lemma 3.4.3 conditional on θ, and then 3.4.1(a) to bound $\|W_{i,x}\|$ and $\|V\|$, we get

$$\left\|\left(\prod_{i=1}^{d} W_{i,x}^{\top}\right)V^{\top}(A^{\top}A - I)V\left(\prod_{i=d}^{1} W_{i,x}\right)x\right\| \le C\varepsilon\|x\|.$$

For $A = I$, this bound is trivial. The given conditions imply also

$$n \geq n_d \geq C'k(\varepsilon^{-1}\log\varepsilon^{-1})\log(n_1 \dots n_d),$$

so applying the same argument with V in place of A yields

$$\left\| \left(\prod_{i=1}^{d} W_{i,x}^\top\right)(V^\top V - I)\left(\prod_{i=d}^{1} W_{i,x}\right)x\right\| \leq C\varepsilon\|x\|.$$

Next, applying Lemma 3.4.1(a–b) yields, for each $j = d, d-1, \dots, 2, 1$,

$$\left\| \left(\prod_{i=1}^{j-1} W_{i,x}^\top\right)(W_{j,x}^\top W_{j,x} - I/2)\left(\prod_{i=j-1}^{1} W_{i,x}\right)x\right\| \leq C\varepsilon\|x\|.$$

Combining these results, we get for the first term of $\nabla f(x)$ that

$$\left\| \left(\prod_{i=1}^{d} W_{i,x}^\top\right)V^\top A^\top AV\left(\prod_{i=d}^{1} W_{i,x}\right)x - 2^{-d}x\right\| \leq C\varepsilon\|x\|. \tag{3.3}$$

This holds with probability at least $1 - e^{-c\varepsilon m} - e^{-c\varepsilon n} - C\sum_i n_i e^{-cn_{i-1}}$.

The second term is controlled similarly: Lemma 3.4.2 implies that for fixed parameters θ, the set $\{V\prod_{i=d}^{j+1} W_{i,x}b_{j,x} : x \in \mathbb{R}^k, j \in [d]\}$ is comprised of at most one of $C(n_1 \dots n_d)^{d(k+1)}$ distinct vectors (which belong to subspaces of dimension 1.) Then applying Lemma 3.4.3 twice to A and V as above, and using also $\|b_{j,x}\| \leq 2$ from Lemma 3.4.1(a),

$$\left\| \left(\prod_{i=1}^{d} W_{i,x}^\top\right)(V^\top A^\top AV - I)\left(\prod_{i=d}^{j+1} W_{i,x}\right)b_{j,x}\right\| \leq C\varepsilon.$$

Applying Lemma 3.4.1(a–b) iteratively as above, we get

$$\left\| \left(\prod_{i=1}^{j} W_{i,x}^\top\right)\left[\left(\prod_{i=j+1}^{d} W_{i,x}^\top\right)\left(\prod_{i=d}^{j+1} W_{i,x}\right) - 2^{-(d-j)}I\right]b_{j,x}\right\| \leq C\varepsilon.$$

Finally, Lemma 3.4.1(a) and (c) yield

$$\left\| \left(\prod_{i=1}^{j} W_{i,x}^\top \right) b_{j,x} \right\| \le C\varepsilon.$$

Combining these, we have for the second term of $\nabla f(x)$ that

$$\left\| \sum_{j=1}^{d} \left(\prod_{i=1}^{d} W_{i,x}^\top \right) V^\top A^\top A V \left(\prod_{i=d}^{j+1} W_{i,x} \right) b_{j,x} \right\| \le C\varepsilon \tag{3.4}$$

also with probability $1 - e^{-c\varepsilon m} - e^{-c\varepsilon n} - C\sum_i n_i e^{-c\varepsilon^2 n_{i-1}}$.

Finally, for the last term of $\nabla f(x)$, if $A \neq I$ then we may apply Lemma 3.4.3 again to get

$$\left\| \left(\prod_{i=1}^{d} W_{i,x}^\top \right) V^\top (A^\top A - I) y \right\| \le C\varepsilon \|y\| \tag{3.5}$$

with probability $1 - e^{-c\varepsilon m}$. Combining (3.3), (3.4), and (3.5) concludes the proof. □

We now bound the second term of g_x.

Lemma 3.4.5. *Under the conditions of Theorem 3.3.1, with probability* $1 - C n_d e^{-c\varepsilon^4 n_{d-1}}$, *for every* $v \in \mathbb{R}^{n_{d-1}}$

$$\left\| W_{d,+,v}^\top V^\top y \right\| \le C\varepsilon \|y\|.$$

Proof. Note that $V^\top y \in \mathbb{R}^{n_d}$ has i.i.d. $\mathcal{N}(0, \|y\|^2/n)$ entries. Then conditional on W_d, for each fixed $v \in \mathbb{R}^{n_{d-1}}$,

$$u(v) \equiv W_{d,+,v}^\top V^\top y \sim \mathbb{N}(0, \Sigma)$$

where

$$\Sigma = (\|y\|^2/n) \cdot W_{d,+,v}^\top W_{d,+,v} \in \mathbb{R}^{n_{d-1} \times n_{d-1}}.$$

On the event that Lemma 3.4.1(b) holds, we have $\|\Sigma\| \le \|y\|^2/n$ and hence $\|u(v)\|^2 \le t n_{d-1} \|y\|^2/n$ with probability $1 - e^{c n_{d-1} t}$ for large t, by a χ^2 tail-bound. Noting that

$n \geq n_d \gg \varepsilon^{-2} n_{d-1}$ and applying this bound for $t = \varepsilon^2 n / n_{d-1}$, we get $\|u(v)\| \leq \varepsilon \|y\|$ with probability $1 - e^{-c\varepsilon^2 n}$.

We use a covering net argument to take a union bound over v: Let N be an ε^2-net of the n_{d-1}-sphere, of cardinality $|N| \leq (3/\varepsilon^2)^{n_{d-1}}$. The above holds uniformly over $v \in N$ with probability $1 - e^{c'\varepsilon^2 n}$, because $n \geq n_d \gg n_{d-1} \cdot \varepsilon^{-2} \log \varepsilon^{-1}$. For any v' on the sphere and $v \in N$ with $\|v - v'\| < \varepsilon^2$, the angle θ between v and v' is at most $C\varepsilon^2$. We have

$$\|u(v) - u(v')\| \leq \left\| W_{d,+,v}^\top - W_{d,+,v'}^\top \right\| \cdot \|V^\top y\|.$$

Suppose now that Lemma 3.4.1(b) holds for W_d with the constant ε^2: This occurs with probability $1 - 8n_d e^{-c\varepsilon^4 n_{d-1}}$. Approximating each of the four terms in

$$\left(W_{d,+,v}^\top - W_{d,+,v'}^\top\right)\left(W_{d,+,v} - W_{d,+,v'}\right)$$

by $I/2$ on this event, we get

$$\left\| W_{d,+,v}^\top - W_{d,+,v'}^\top \right\|^2 = \left\| \left(W_{d,+,v}^\top - W_{d,+,v'}^\top\right)\left(W_{d,+,v} - W_{d,+,v'}\right) \right\| \leq C'(\varepsilon^2 + \theta) \leq C\varepsilon^2.$$

Thus on this event, $\|u(v) - u(v')\| \leq C\varepsilon \|V^\top y\|$. By a χ^2 tail-bound, with probability $1 - e^{-cn_d}$ we have $\|V^\top y\|^2 \leq 2\|y\|^2 n_d / n \leq 2\|y\|^2$ and hence $\|u(v) - u(v')\| \leq C\varepsilon \|y\|$. □

Proof of Theorem 3.3.1. Combining Lemmas 3.4.4, 3.4.5, and 3.4.1(a), with the stated probability,

$$\|\nabla f(x) - 2^{-d} x\| \leq C\varepsilon(1 + \|x\| + \|y\|)$$

for every $x \in \mathbb{R}^k$. Since G is piecewise linear, the directional derivative $D_v f(x)$ always exists at any $x \in \mathbb{R}^k$ for any unit vector $v \in \mathbb{R}^k$, even for x where f is non-differentiable. Set $\tilde{x} = x/\|x\|$. For any fixed x, there exists a sequence $\{x_n\}$ which converges to x and where f is differentiable, such that

$$D_{-\tilde{x}} f(x) = \lim_{n \to \infty} -\tilde{x}^\top \nabla f(x_n)$$

Since

$$-\tilde{x}^\top \nabla f(x_n) = -2^{-d} \tilde{x}^\top x_n + \tilde{x}^\top (2^{-d} x_n - \nabla f(x_n)) \leq -2^{-d} \tilde{x}^\top x_n + C\varepsilon(1 + \|x_n\| + \|y\|),$$

we get

$$\begin{aligned} D_{-\tilde{x}} f(x) &\leq \liminf_{n \to \infty} \Big[-2^{-d} \tilde{x}^\top x_n + C\varepsilon(1 + \|x_n\| + \|y\|) \Big] \\ &= -2^{-d} \|x\| + C\varepsilon(1 + \|x\| + \|y\|). \end{aligned}$$

For $\varepsilon > 0$ sufficiently small and $C' > 0$ sufficiently large, this implies $D_{-\tilde{x}} f(x) < 0$ whenever $\|x\| \geq C'\varepsilon(1 + \|y\|)$. $\square$

3.5 Experiments

We present experiments to illustrate the performance of surfing over a sequence of networks during training compared with gradient descent over the final trained network. We mainly use the Fashion-MNIST dataset[1] to carry out the simulations, which is similar to MNIST in many characteristics, but is more difficult to train. We build multiple generative models, trained using VAE (Kingma and Welling, 2014), DCGAN (Radford et al., 2015), WGAN (Arjovsky et al., 2017) and WGAN-GP (Gulrajani et al., 2017). The structure of the generator/decoder networks that we use are the same as those reported by Chen et al. (2016); they include two fully connected layers and two transposed convolution layers with batch normalization after each layer

1. https://github.com/zalandoresearch/fashion-mnist

Input dimension		5	10	20	5	10	20
	Model	VAE			DCGAN		
% successful	Regular Adam	98.7	100	100	48.3	68.7	80.0
	Surfing	100	100	100	78.3	98.7	96.3
# iterations	Regular Adam	737	1330	8215	618	4560	18937
	Surfing	775	1404	10744	741	6514	33294
	Model	WGAN			WGAN-GP		
% successful	Regular Adam	56.0	84.3	90.3	47.0	64.7	64.7
	Surfing	81.7	97.3	99.3	83.7	95.7	97.3
# iterations	Regular Adam	464	1227	3702	463	1915	15445
	Surfing	547	1450	4986	564	2394	25991

Table 3.1: Surfing compared against direct gradient descent over the final trained network, for various generative models with input dimensions $k = 5, 10, 20$. Shown are percentages of “successful” solutions $\hat{x}_T$ satisfying $\|\hat{x}_T - x_*\| < 0.01$, and 75th-percentiles of the total number of gradient descent steps used (across all networks $G_0, \ldots, G_T$ for surfing) until $\|\hat{x}_T - x_*\| < 0.01$ was reached.

(Ioffe and Szegedy, 2015). We use the simple surfing algorithm in these experiments, rather than the projected-gradient algorithm proposed for theoretical analysis. Note also that the network architectures do not precisely match the expansive relu networks used in our analysis. Instead, we experiment with architectures and training procedures that are meant to better reflect the current state of the art.

We first consider the problem of minimizing the objective $f(x) = \frac{1}{2}\|G(x) - G(x_*)\|^2$ and recovering the image generated from a trained network $G(x) = G_{\theta_T}(x)$ with input x_*. We run surfing by taking a sequence of parameters $\theta_0, \theta_1, ..., \theta_T$ for $T = 100$, where θ_0 are the initial random parameters and the intermediate θ_t's are taken every 40 training steps, and we use Adam (Kingma and Ba, 2014) to carry out gradient descent in x over each network G_{θ_t}. We compare this to “regular Adam”, which uses Adam to optimize over x in only the final trained network G_{θ_T} for $T = 100$.

To ensure that the runtime of surfing is comparable to that of a single initialization of regular Adam, we do not run Adam until convergence for each intermediate network in surfing. Instead, we use a fixed schedule of iterations for the networks

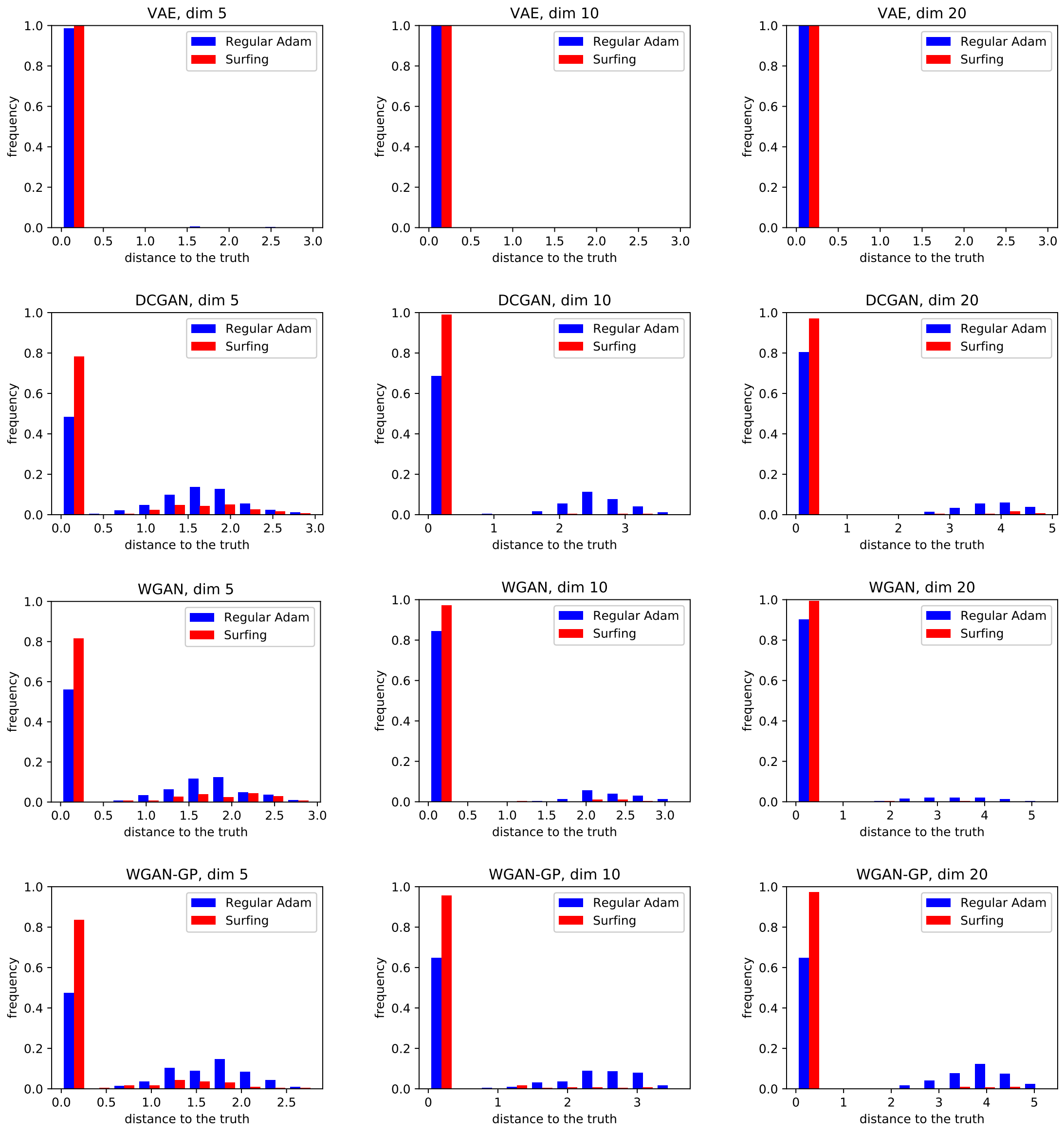


Figure 3.2: Distribution of distance between solution $\hat{x}_T$ and the truth x_* for VAE, DCGAN, WGAN and WGAN-GP, comparing surfing (red) to regular gradient descent (blue) over the final network.

$G_{\theta_0}, \ldots, G_{\theta_{T-1}}$, and run Adam to convergence in only the final network G_{θ_T}. The total number of iterations for networks $G_{\theta_0}, \ldots, G_{\theta_{T-1}}$ is set as the 75th-percentile of the iteration count required for convergence of regular Adam. These are split across the networks proportional to a deterministic schedule that allots more steps to the

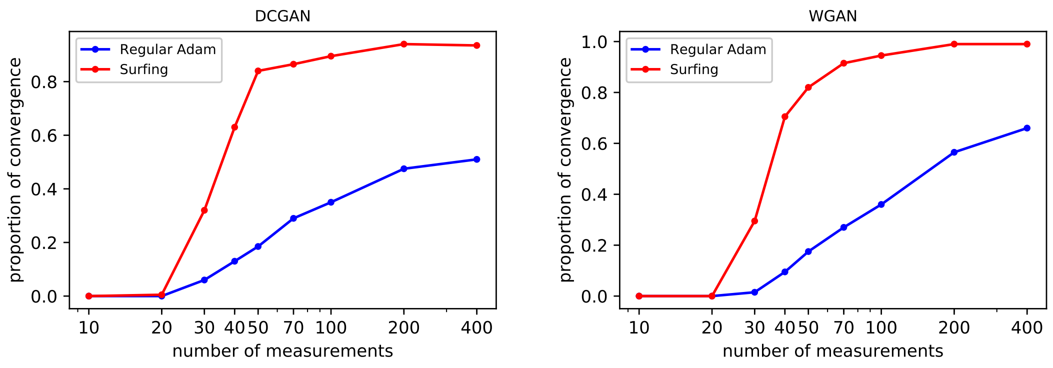


Figure 3.3: Compressed sensing setting for exact recovery. As a function of the number of random measurements m, the lines show the proportion of times surfing (red) and regular gradient descent with Adam (blue) are able to recover the true signal $y = G(x)$, using DCGAN and WGAN.

earlier networks where the landscape of $G(x)$ changes more rapidly, and fewer steps to later networks where this landscape stabilizes.

For each network training condition, we apply surfing and regular Adam for 300 trials, where in each trial a randomly generated x_* and initial point x_{init} are chosen uniformly from the hypercube $[-1, 1]^k$. Table 3.1 shows the percentage of trials where the solutions $\hat{x}_T$ satisfy our criterion for successful recovery $\|\hat{x}_T - x_*\| < 0.01$, for different models and over three different input dimensions k. The table also shows the 75th-percentile for the total number of gradient descent iterations taken (across all networks for surfing), verifying that the runtime of surfing was typically 1–2x that of regular Adam.

We also provide the distributions of $\|\hat{x}_T - x_*\|$ under each setting: Figure 3.2 shows the results for VAE, DCGAN, WGAN and WGAN-GP. The results indicate that direct descent often succeeds, but can also converge to a point that is far from the optimum. By moving along the optimum of the evolving surface, surfing is able to move closer to the optimum in these cases.

We next consider the compressed sensing problem with objective $f(x) = \frac{1}{2}\|AG(x) -$

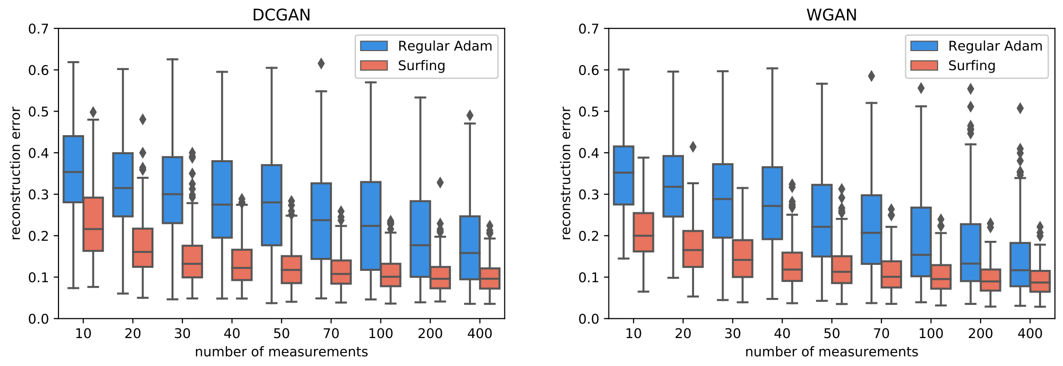


Figure 3.4: Compressed sensing setting for approximation, or rate-distortion. As a function of the number of random measurements m, the box plots summarize the distribution of the per-pixel reconstruction errors for DCGAN and WGAN trained models, using surfing (red) and regular gradient descent with Adam (blue).

$AG(x_*)\|^2$ where $A \in \mathbb{R}^{m\times n}$ is the Gaussian measurement matrix. We carry out 200 trials for each choice of number of measurements m. The parameters θ_t for surfing are taken every 100 training steps. As before, we record the proportion of the solutions that are close to the truth x_* according to $\|\hat{x}_T - x_*\| < 0.01$. Figure 3.3 shows the results for DCGAN and WGAN trained networks with input dimension $k = 20$.

Lastly, we consider the objective $f(x) = \frac{1}{2}\|AG(x) - Ay\|^2$, where y is a real image from the hold-out test data. This can be thought of as a rate-distortion setting, where the error varies as a function of the number of measurements used. We carry out the same experiments as before and compute the average per-pixel reconstruction error $\sqrt{\frac{1}{n}\|G(\hat{x}_T) - y\|^2}$ as in Bora et al. (2017). Figure 3.4 shows the distributions of the reconstruction error as the number of measurements m varies.

3.6 Discussion

In this chapter, we has explored the idea of incrementally optimizing a sequence of objective risk functions obtained from models that are slowly changing during stochastic gradient descent during training. When initialized with random parameters

θ_0, we have shown that the empirical risk function $f_{\theta_0}(x) = \frac{1}{2}\|G_{\theta_0}(x) - y\|^2$ is well behaved and easy to optimize. The surfing algorithm initializes x for the current network $G_{\theta_t}(x)$ at the optimum x^*_{t-1} found for the previous network $G_{\theta_{t-1}}(x)$ and then carries out gradient descent to obtain the updated point $x^*_t = \text{argmin}_x f_{\theta_t}(x)$. Our experiments show that this scheme has merit, and often significantly outperforms direct gradient descent on the final model alone.

On the theoretical side, our main technical result applies and extends ideas of Hand and Voroninski (2019) to show that for random ReLU networks that are sufficiently expansive, the surface of $f_{\theta_0}(x)$ is well-behaved for arbitrary target vectors y. This result may be of independent interest, but it is essential for the surfing algorithm because initially the model is poor, with high approximation error.

The analysis for the incremental scheme uses projected gradient descent, although we find that simple gradient descent works well in practice. The analysis assumes that the argmin over the surface evolves continuously in training. This assumption is necessary—if the global minimum is discontinuous as a function of t, so that the minimizer "jumps" to a far away point, then the surfing procedure will fail in practice.

3.7 Future Work

In our experiments, we see that simple surfing can indeed be effective for mapping outputs y to inputs x for the trained network, where it often outperforms direct gradient descent for a range of deep network architectures and training procedures. However, these simulations also point to the fact that in some settings, direct gradient descent itself can be surprisingly effective. A deeper understanding of this phenomenon could lead to more advanced surfing algorithms that are able to ride to the final optimum even more efficiently and often.

Beside inverting a generative network and solving a compressed sensing problem, the idea of surfing can also be applied to solve other optimization problems whose objective involves a trained machine learning model. Specifically, when minimizing a nonconvex objective $f(x; M_\theta)$ that involves a model M_θ parameterized by θ, surfing can in principle be applied to a sequence of objectives $f(x; M_{\theta_t})$ obtained from training M_θ. If the landscape of $f(x; M_{\theta_t})$ is nice at initialization and evolves continuously as θ_t is trained, we can expect a better solution with surfing than direct gradient descent on the final trained model $f(x; M_{\theta_T})$. An immediate extension of surfing could be to phase retrieval where the signals are from a generative prior (Hand et al., 2018).

Chapter 4

Translation between Brain Modalities with Artificial Neural Networks

Artificial neural networks (ANNs) have achieved tremendous success in various areas, including computer vision, natural language processing, motion control and healthcare. ANNs are computational graphs inspired from neural networks in the brain. The basic elements of ANN are neurons, which connect to each other according to specific structures. By varying the structure of the network, an ANN is able to adapt to different tasks.

Although ANNs are designed by analogy to circuits in the brain, the actual mechanism of the brain still remains unclear. In this chapter, we try to take advantage of modern ANN models to analyze brain imaging data. In particular, we build proper ANN models to explore the relationship between the electrical signals measured through fluorescence Ca^{2+} imaging, and blood oxygen level dependent (BOLD) signals measured through functional magnetic resonance imaging (fMRI). Given two modalities are collected simultaneously in mice, we build a translation model to predict the BOLD signals from the calcium. We also show the brain connectivity map has been preserved in the predictions. The experimental results indicate that our models have the power of translate from one type of signal to another and demonstrate the merit of ANNs in helping understand brain activity.

4.1 Introduction

ANN models have been widely used for computer vision tasks since convolutional neural networks (CNN) were proposed by LeCun et al. (1995) as a supervised classification model. The ResNet architecture (He et al., 2016) adds skip connections to CNNs that allow for a deeper network without the vanishing gradient problem, and further improves model performance on image classification tasks. On the other hand, generative adversarial networks (GANs) (Goodfellow et al., 2014) provide a fundamental training framework for unsupervised image generation tasks. The model contains a generator and a discriminator network that are trained simultaneously in an adversarial manner — the generator aims to fool the discriminator by generating fake images while the discriminator aims to differentiate them from real images from the training set. The training objective is then to solve a minimax problem where the solution is a saddle point of the loss function, which makes it difficult for the gradient descent algorithm to converge. DCGANs (Radford et al., 2015) incorporate convolutional structures into both the generator and discriminator, and stabilizes the training with batch normalization (Ioffe and Szegedy, 2015) and comprehensive parameter tuning. WGANs (Arjovsky et al., 2017) use Wasserstein distance as the training loss to measure the difference between the distributions of generated images and real images, which also stabilizes the training and improve the quality of generated images. During training of a WGAN, the parameters in the discriminator have to be clipped after every gradient step in order to maintain the Lipschitz property of the network. Alternatively, WGAN-GP (Gulrajani et al., 2017) introduces a gradient penalty term into the loss function and gets rid of the unnatural weight clipping operation. PGGAN (Karras et al., 2017) and Style GAN (Karras et al., 2019) are recent advances in image generation based on GANs that are able to achieve state-of-the-art performance on generating high-resolution images.

The adversarial training framework is also often used in image translation tasks. Similar to language translation, the image translation model takes an image as input and outputs an image that has a certain relation with the input. Typical image translation problems include style change and semantic segmentation. Conditional GANs (`pix2pix`) (Isola et al., 2017) use the GAN framework to solve the supervised image-to-image translation problem, where the training data contains paired source and target images. A conditional GAN combines the GAN loss with the reconstruction loss in its training objective and uses a "U-Net" structure for its generator that enables extraction of low-level features of images while preserving the high-level features. In the unsupervised settings where only the marginal distribution of source and target images are given or the images are unpaired, the Cycle-GAN framework (Zhu et al., 2017a) considers training a source-to-target GAN and a target-to-source GAN at the same time, and its training objective consists of the two GAN losses for both directions and a cycle-consistent loss. UNIT (Liu et al., 2017) trains two GANs to generate images from source and target distributions respectively but with a shared latent space. Two encoder networks are also trained at the same time to map the images back to the common latent space. Since the encoder and generator also form a VAE model, the training objective consists of two VAE losses, two GAN losses and cycle-consistent losses. Bicycle-GAN (Zhu et al., 2017b) considers multimodal translation problem where multiple target images can be generated from a single source image. The model is still based on conditional GANs but introduces an encoder to map the target image to a Gaussian latent space for a style variable.

The generator takes both the input image and the style variable as input to produce an output image. The training loss is a hybrid of the losses of the encoder-generator model (cVAE-GAN) and generator-encoder model (cLR-GAN).

In this chapter, we develop tools for working with two brain imaging modalities, the whole-brain blood oxygen level-dependent (BOLD) imaging and cortex-wide fluorescence Ca^{2+} imaging. The BOLD signals are collected through functional magnetic resonance imaging (fMRI), which provides a non-invasive measure of activity with whole-brain coverage. However, fMRI is limited by relatively low spatial and temporal resolutions and low signal-to-noise ratios (SNR). On the other hand, Ca^{2+} imaging is able to examine the concerted activity of neurons across a large field of view (FOV) with fairly high resolution and SNR, but the signal collection requires invasive manipulation of the nervous system and is only limited to optically accessible tissue, *i.e.*, cerebral cortex. Lake et al. (2020) combine Ca^{2+} and fMR imaging techniques together and acquire the first simultaneous cortex-wide calcium imaging and whole-brain fMRI in mice. In collaboration with their group, we apply modern ANN models on this concurrent calcium/BOLD data set. In particular, we build translation models that predict global (whole-brain) BOLD signals from local (cortex-wide) calcium signals, and the experimental results suggest that the model is able to capture brain functional mechanism and fill in the missing information of the modality.

The rest of this chapter is organized as follows. We first briefly describe the data acquisition process together with the data pre-processing in Section 4.2. Then, as a preliminary computational experiment, we show in Section 4.3 that with the WGAN-GP architecture, we are able to synthesize calcium and BOLD images of high quality. In Section 4.4, we implement a conditional GAN to translate from calcium images to BOLD. However, we find that the model is not able to give a favorable prediction for normalized BOLD images because of the low SNR of fMRI. Therefore, in Section 4.5 we instead consider a simplified data set where the signals are averaged in each of the brain regions-of-interest (ROI). We show our translation model can preserve the brain connectivity map by comparing the connectivity matrix of predicted BOLD sequences with the truth. Finally, we discuss our results and future research directions in Section 4.6.

4.2 Data Acquisition and Pre-processing

4.2.1 Data Acquisition

The calcium images and BOLD images are simultaneously acquired by imaging mice using a special approach (Lake et al., 2020). In the experiments, each mouse is imaged for three sessions. Each session contains seven 10-minute runs, and in some of the runs the mouse receives an LED stimulus in its eyes. There are 10 mice involved in the experiments while the image data is available for three of them.

The calcium images are recorded in cerebral cortex with interleaved violet (Ca^{2+}-insensitive) and cyan (Ca^{2+}-sensitive) illumination, at a rate of 20Hz. Both wavelengths are smoothed, motion corrected and downsampled by a factor of two. In order to remove the background noise, the violet wavelength is regressed from the cyan pixel-wise. The final calcium images are grey-level 2D images at 10 frames per second, with a shape of 390×390 pixels.

The BOLD data are collected through whole-brain functional magnetic resonance imaging (fMRI) with a 1Hz sampling rate. The data are also processed with motion correction and denoising procedures. The final BOLD images are grey-level 3D images with a shape of $64 \times 32 \times 28$.

In order to study the functions of different brain regions, calcium and BOLD images are co-registered using the vascular anatomy in the cortex as landmarks. Then, the Allen mouse brain atlas is referenced to identify the brain regions of interest (ROIs) for both calcium and BOLD images. Further details of data acquisition and registration can be found in (Lake et al., 2020).

4.2.2 Data Pre-processing

Although the image data has been processed after it is collected from the medical devices, it still needs further manipulation before being ready to feed into neural network models. The calcium images are first centered and rotated so they have the same orientation. We cut out the part that are outside the brain, giving 190×190 pixel images, which then are resized to 128×128 for the convenience of model-building. To avoid extreme pixel values, we Winsorize the pixel values according their 0.025% and 99.975% quantiles. Specifically, we apply the following Winsorization function on each pixel:

$$w(x) = \begin{cases} q_1, & x \leq q_1, \\ x, & q_1 < x \leq q_2, \\ q_2, & x > q_2, \end{cases}$$

where q_1 and q_2 are the 0.025% and 99.975% percentiles. Finally, we move the pixel values into $[-1, 1]$ through a linear transformation.

The BOLD images have the shape $64 \times 32 \times 28$. We center the brain in the images and trim them to $32 \times 32 \times 16$. The pixel values are also Winsorized and transformed into $[-1, 1]$ following the same procedure as for the calcium images. Samples of calcium and BOLD images after pre-processing can be found in Fig. 4.1a and Fig. 4.2.

4.3 Image Generation with GANs

Generative adversarial networks (GANs) are unsupervised ANN models used to generate samples from a target distribution (Goodfellow et al., 2014). The distributions that we want to sample from are usually high-dimensional and difficult to model with classical statistical models. GANs consist of a generator and a discriminator that are trained in an adversarial fashion, where the generator aims to generate fake data to fool the discriminator, while the goal of the discriminator is to differentiate them from the real data. In this section, we describe a variant of GAN used to obtain synthetic calcium and BOLD images.

The objective of GAN training is to solve a minimax problem. Given the generator G and discriminator D, it can be written as

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[1 - \log D(G(z))],$$

where p_z is usually assumed to be standard Gaussian or uniform distribution. The Wasserstein GAN or WGAN (Arjovsky et al., 2017) variant uses Warsserstein distance to measure the difference between p_G, the distribution of samples generated from G, and p_{data}, the target distribution, with the training objective given by

$$\min_G \max_{D:\|D\|_L\leq 1} \mathbb{E}_{x\sim p_{\text{data}}}[D(x)] - \mathbb{E}_{z\sim p_z}[D(G(z))].$$

When training a WGAN, a weight clipping operation on D should be added after each SGD update to ensure the Lipschitz constant of D is always upper bounded. Alternatively, WGAN-GP (Gulrajani et al., 2017) enforces the Lipschitz constant by adding a gradient penalty on the objective, according to

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}}[D(x)] - \mathbb{E}_{z\sim p_z}[D(G(z))] + \lambda \mathbb{E}_{\hat{x}\sim p_{\hat{x}}}\left[(\|\nabla D(\hat{x})\|_2 - 1)^2\right], \tag{4.1}$$

where $p_{\hat{x}} = (1-u)p_G + up_{\text{data}}$ with $u\sim$Uniform$[0,1]$.

Calcium image generation. We first train a WGAN-GP model for the calcium image generation task. Table 4.1 gives the architectures of the generator and dis-

Generator G

Layer	Output shape
Input	128
FC, batchnorm, ReLU	$4 \times 4 \times 512$
Upconv, batchnorm, ReLU	$8 \times 8 \times 512$
Upconv, batchnorm, ReLU	$16 \times 16 \times 256$
Upconv, batchnorm, ReLU	$32 \times 32 \times 128$
Upconv, batchnorm, ReLU	$64 \times 64 \times 64$
Upconv, tanh	128×128

(a) Generator network.

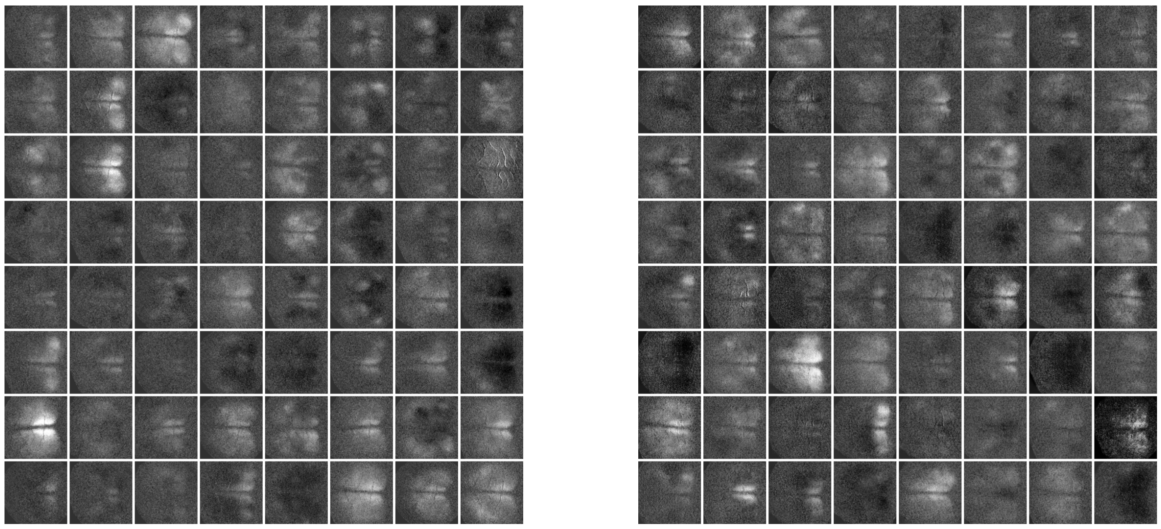
Discriminator D

Layer	Output shape
Input	128×128
Conv, leakyReLU	$64 \times 64 \times 32$
Conv, layernorm, leakyReLU	$32 \times 32 \times 64$
Conv, layernorm, leakyReLU	$16 \times 16 \times 128$
Conv, layernorm, leakyReLU	$8 \times 8 \times 256$
Conv, layernorm, leakyReLU	$4 \times 4 \times 512$
FC	1

(b) Discriminator network.

Table 4.1: Architecture of WGAN-GP on calcium images

criminator networks in the WGAN-GP model, where conv and upconv denote the convolution and transposed convolution operation with a 4×4 kernel and stride 2. We pick 2000 out of 6000 images in each 10-minute run and use the calcium images for all three mice, which forms a training set of 126,000 images (3 mice $\times$ 3 sessions $\times$ 7 runs $\times$ 2000 images). The model is trained for 50 epochs where the batch size is 64. On each mini batch, the discriminator is trained for 5 steps before the generator is trained for one step. The initial learning rates for both the generator and discriminator are 0.001. Then, both learning rates are divided by 3 after every 10 epochs that allows the training from coarse to fine. The gradient penalty coefficient λ in (4.1) is set to 10. Fig. 4.1 compares the real calcium images with images generated from the WGAN-GP; the images in both panels are randomly sampled from the data set



(a) Real calcium images (b) Synthetic calcium images

Figure 4.1: Comparison between real (left) and synthetic (right) calcium images by WGAN-GP

or the model. We observe that the synthetic images are similar to the real ones. In particular, the synthetic calcium signals are bilaterally symmetric on cortex surface, and the vasculature on the brain can also be observed from some of the synthetic images.

BOLD images generation. The BOLD images are 3D tensors with dimension $32 \times 32 \times 16$. We train WGAN-GPs with both 2D and 3D convolutional networks. In the 2D network, we regard the 3D images as 32×32 images with 16 channels. The kernels used in the convolutional layers are 4×4. In the 3D network, since the outputs of each layer are 4-dimensional tensors (disregarding the dimension on batch size) with shape [height, width, depth, channel], we use $4 \times 4 \times 4$ kernels for the convolution. The architectures of the generator and discriminator networks with both 2D and 3D convolutions are summarized in Table 4.2. We use all the BOLD images of the three mice as the training data, so it contains 37,800 images in total (3 mice $\times$ 3 sessions $\times$ 7 runs $\times$ 600 images). We adopt the same training parameters as for calcium data, except that the learning rate is divided by 2 after every 10 epochs.

Generator G

Layer	Output shape (2D)	Output shape (3D)
Input	128	128
FC, batchnorm, ReLU	$4 \times 4 \times 1024$	$4 \times 4 \times 2 \times 512$
Upconv, batchnorm, ReLU	$8 \times 8 \times 1024$	$8 \times 8 \times 4 \times 256$
Upconv, batchnorm, ReLU	$16 \times 16 \times 512$	$16 \times 16 \times 8 \times 128$
Upconv, tanh	$32 \times 32 \times 16$	$32 \times 32 \times 16$

(a) Generator networks.

Discriminator D

Layer	Output shape (2D)	Output shape (3D)
Input	$32 \times 32 \times 16$	$32 \times 32 \times 16$
Conv, leakyReLU	$16 \times 16 \times 256$	$16 \times 16 \times 8 \times 128$
Conv, layernorm, leakyReLU	$8 \times 8 \times 512$	$8 \times 8 \times 4 \times 256$
Conv, layernorm, leakyReLU	$4 \times 4 \times 1024$	$4 \times 4 \times 2 \times 512$
FC	1	1

(b) Discriminator networks.

Table 4.2: Architecture of WGAN-GP with 2D and 3D convolutions on BOLD images

Figs. 4.3 and 4.4 present samples of synthetic BOLD images from the 2D and 3D models respectively. We observe that the synthetic images from both models are of high quality and very similar to the real BOLD images in Fig. 4.2. The anatomy at each layer of the brain can be clearly observed from the synthetic images.

4.4 Image Translation with Conditional GANs

In this section, we focus on the problem of predicting BOLD images from contemporaneous calcium images. We use a conditional GAN (`pix2pix`) model (Isola et al., 2017) which is designed for image-to-image translation tasks. Similar to GAN, it consists of a generator G and discriminator D that are trained adversarially by

$$\min_G \max_D \mathcal{L}_{\text{GAN}}(G, D) + \mu \mathcal{L}_{\text{L1}}(G)$$

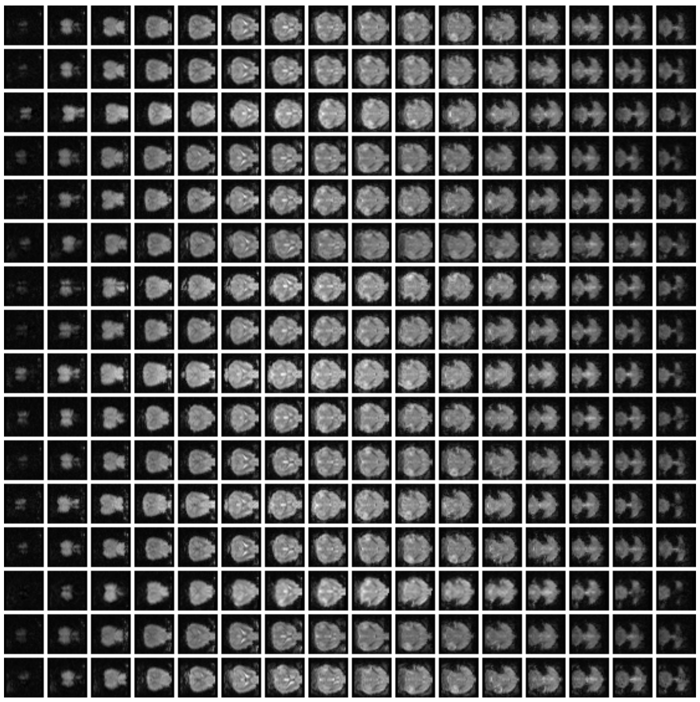


Figure 4.2: BOLD images after pre-processing, where each row presents randomly sampled $32 \times 32 \times 16$ BOLD images in 16 slices.

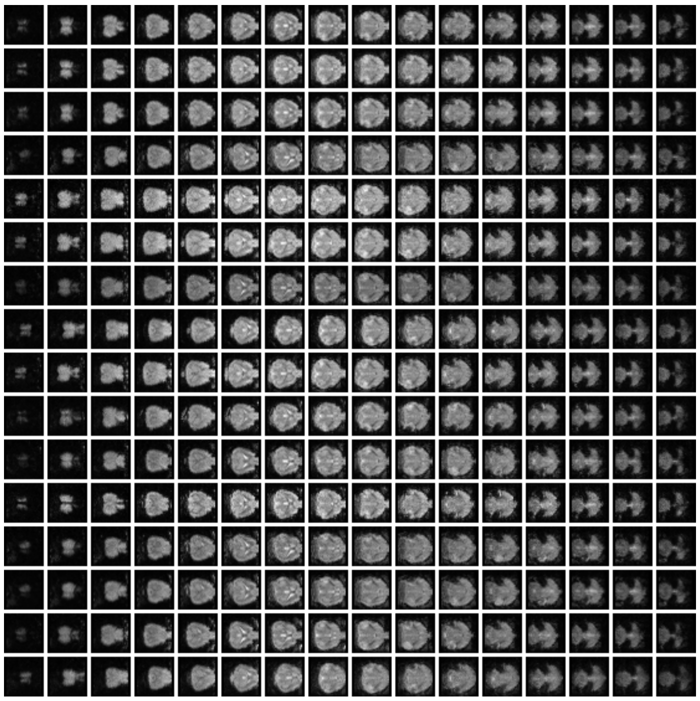


Figure 4.3: Synthetic BOLD images randomly sampled from a trained WGAN-GP with 2D convolutions.

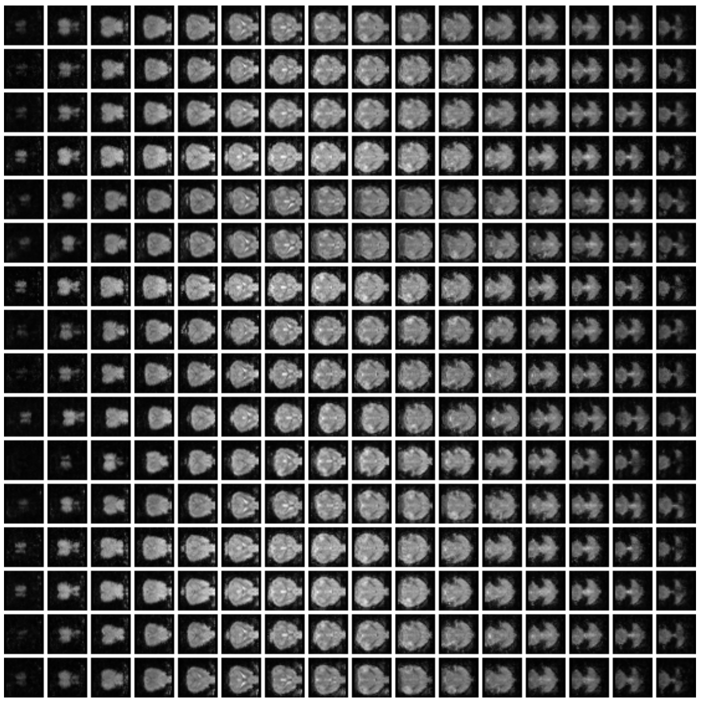


Figure 4.4: Synthetic BOLD images randomly sampled from a trained WGAN-GP with 3D convolutions.

where the GAN loss is

$$\mathcal{L}_{\text{GAN}}(G, D) = \mathbb{E}_{y \sim p_y}[\log D(y)] + \mathbb{E}_{x \sim p_x}[1 - \log D(G(x))] \tag{4.2}$$

and the L1 loss is

$$\mathcal{L}_{\text{L1}}(G) = \mathbb{E}_{x,y \sim p_{x,y}} \|y - G(x)\|_1. \tag{4.3}$$

Note that the loss involves $p_{x,y}$, the joint distribution of input images x and target images y, which means training requires paired images (x_i, y_i); thus it is a supervised algorithm. To improve the performance of the model, we use the WGAN-GP loss in place of the GAN loss $\mathcal{L}_{\text{GAN}}(G, D)$, *i.e.*

$$\mathcal{L}_{\text{WGAN-GP}}(G, D) = \mathbb{E}_{y \sim p_y}[D(y)] - \mathbb{E}_{x \sim p_x}[D(G(x))] + \lambda \mathbb{E}_{\hat{y} \sim p_{\hat{y}}}\left[(\|\nabla D(\hat{y})\|_2 - 1)^2\right]$$

where $p_{\hat{y}}$ is defined similarly as $p_{\hat{x}}$ in (4.1).

Generator G

Layer	Output shape
Input	128×128
Downsample	$64 \times 64 \times 64$
Downsample [1]	$32 \times 32 \times 128$
Downsample [2]	$16 \times 16 \times 256$
Downsample [3]	$8 \times 8 \times 512$
Downsample	$4 \times 4 \times 1024$
Upsample, dropout, concat [3]	$8 \times 8 \times 1024$
Upsample, dropout, concat [2]	$16 \times 16 \times 512$
Upsample, concat [1]	$32 \times 32 \times 256$
Conv, tanh	$32 \times 32 \times 16$

(a) Generator network.

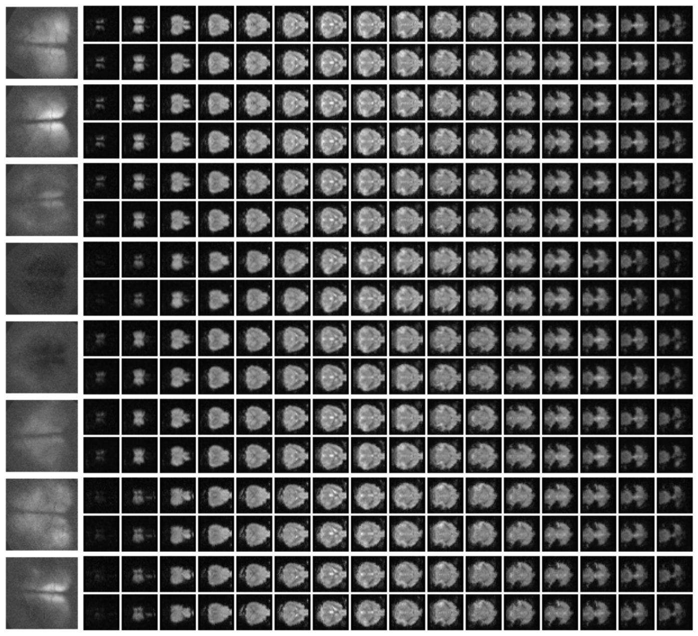
Discriminator D

Layer	Output shape
Input	$32 \times 32 \times 16$
Conv, layernorm leakyReLU	$16 \times 16 \times 256$
Conv, layernorm, leakyReLU	$8 \times 8 \times 512$
Conv, layernorm, leakyReLU	$4 \times 4 \times 1024$
FC	1

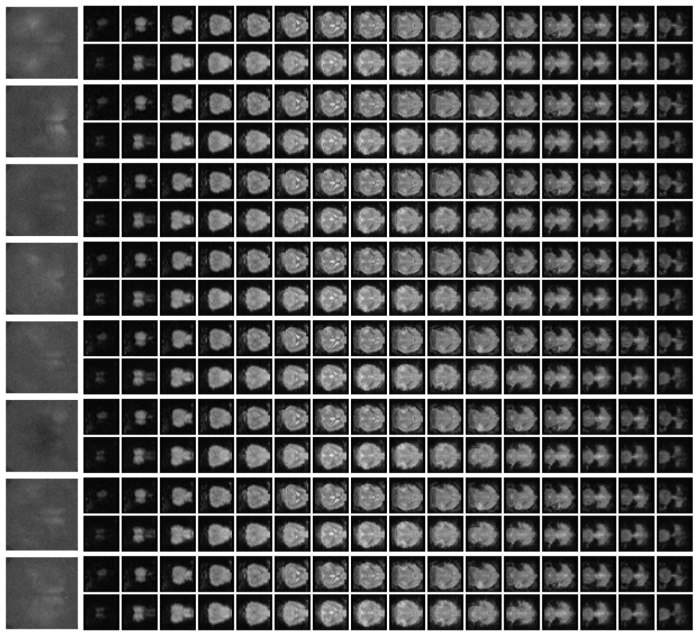
(b) Discriminator network.

Table 4.3: Architecture of `pix2pix` model. Downsampling block includes a convolutional layer followed by batch normalization and ReLU activation, while upsampling block composes of transposed convolution, batch normalization and leakyReLU.

Notice that it is not a standard image-to-image translation problem, given that the inputs are 2D calcium images while the targets are 3D BOLD images. In order to reconcile the discrepancy in dimension, we treat the 3D BOLD images as 2D images with multiple channels and apply 2D convolutions in our networks. Since there is spatial correspondence between the calcium images and each slice of the BOLD images, we also adopt the "U-Net" structure for the generator, following Isola et al. (2017). Specifically, the "U-Net" consists of an encoder and a decoder network, where the encoder network is able to extract the high-level features of input images through a series of down-sampling convolutional layers, and the decoder network generates target



(a) Comparison between real and predicted BOLD images on training data.



(b) Comparison between real and predicted BOLD images on test data.

Figure 4.5: Predicting BOLD images from calcium images with `pix2pix` model. For each of the eight rows in a figure, the input calcium image is on the left, the real 3D BOLD image is presented at the top sub-row on the right, and the predicted BOLD image is below the real one at the bottom sub-row.

images from those features with up-sampling layers. There are also skip connections between each pair of layers in encoder and decoder networks that have the same size of feature map. These skip connections circumvent the bottleneck in the middle and allow low-level features to shortcut across the network. In addition, we also include dropout layers with dropout rate 0.5 in the generator to avoid over-fitting. The detailed architecture of the `pix2pix` model for our translation experiments is presented in Table 4.3.

A study by Ma et al. (2016) shows that BOLD signals in the brain always have approximately a 1.4 second delay compared to calcium signals, since the blood oxygenation is a secondary metabolic process. During training, we use image pairs (x, y) where x is the average of 5 consecutive calcium images 1.4 second prior to the target BOLD image y. We train the model for 100 epochs. The learning rate is 0.0001 at beginning and divided by 2 after every 20 epochs. The gradient penalty coefficient λ is still 10 and the L1 loss coefficient μ is set to 100 as suggested by Isola et al. (2017).

Fig. 4.5 shows the results of predicting BOLD images from calcium images with the `pix2pix` model on both the training set and the test set. Recall that the images are collected on three mice with three sessions. We select data from all three sessions for the first mouse and two sessions for the second mouse as the training set and use the rest as the test set. In each figure of Fig. 4.5, we show eight groups of images in rows. For each of the eight rows, the input calcium image is on the left, the real 3D BOLD image is presented at the top sub-row on the right, and the predicted BOLD image is below the real one at the bottom sub-row. We observe that the model can well predict the BOLD images on training data, while on the test data that contains data from a different mouse, it can still generate reasonable BOLD images, but the predicted images have less resemblance to the real ones. In fact, compared with calcium images, the BOLD images have much smaller variance for each mouse. Thus, the model can capture the anatomical features of the mouse brain, but it does not capture the functional variations of the BOLD images.

In order to address this problem, we normalize the BOLD images to remove the anatomical information. Specifically, for each pixel of the BOLD images, we subtract the mean of its pixel value across a whole session for each mouse. We don't subtract mean across each 10-minute run because there are several runs in the session where the mouse receives LED stimulus, and normalizing for each run would potentially damage this information. We retrain the model with the normalized BOLD images and the results are presented in Fig. 4.6. We observe that the model can now fit the normalized BOLD images to some extent on training data, but fails to give reasonable predictions on test data. The pixel values on predicted BOLD images are close to zero, which indicates the model has difficulty predicting the BOLD signals from the calcium images that it has not seen before. One reason why the model fails might be that the signal-to-noise ratio in BOLD images, compared with calcium images, is low. The BOLD images capture the blood oxygen level signals of the whole brain, and result from many other factors than the calcium concentration on cerebral cortex. The relationship between these two signals might be especially weak at deep layers of the brain. Unlike the calcium images, the dynamics of the BOLD is not easy to view by eye, even after normalization. Considering that there is also noise introduced from the measurements, the BOLD dynamics therefore has a faint signal with respect to calcium level, which causes the weak model predictions on test data.

4.5 Translation Models on ROI Data

The above analysis indicates that the low signal-to-noise ratio on fMRI data prevents us from obtaining favorable translation results. In order to reduce the problem complexity and noise level, we turn to modeling with a simplified data set, where the high-dimensional calcium and BOLD images are reduced to low-dimensional time series. Specifically, after image registration with the Allen atlas, we are able to divide the whole brain into a number of regions-of-interest (ROIs). The calcium and BOLD signal vectors are computed by taking the mean of the pixel values of the calcium or BOLD images within the same ROI. Given the calcium signals are relatively consistent within each second, we down-sample the signals from 10 Hz to 1 Hz so that it matches the frequency of the BOLD data.

Generator G

Layer	Output shape
Input	64×44
Downsample [1]	32×128
Downsample [2]	16×256
Downsample [3]	8×512
Downsample	4×1024
Upsample, dropout, concat [3]	8×1024
Upsample, dropout, concat [2]	16×512
Upsample, dropout, concat [1]	32×256
Upsample	64×128
Conv, batchnorm, tanh	64×44

(a) Generator network.

Discriminator D

Layer	Output shape
Input	64×44
Conv, layernorm leakyReLU	32×128
Conv, layernorm, leakyReLU	16×356
Conv, layernorm, leakyReLU	8×512
FC	1

(b) Discriminator network.

Table 4.4: Architecture of `pix2pix` model on ROI data. Downsampling block includes a convolutional layer followed by batch normalization and ReLU activation, while upsampling block composes of transposed convolution, batch normalization and leakyReLU. The dropout rate is 0.5.

We still focus on the problem of predicting BOLD signals from calcium. Since the data is available in 44 ROIs, the input of the model is the 44-dimensional calcium time series of length 64 seconds and the output is the BOLD time series within the same time period. There are several differences in data format between this problem and the previous one that requires a change in the design of model structure. In the image translation problem, the data are 2D/3D images and a single BOLD image is predicted from an averaged calcium image at a given time, while in current setting, the data are time series of vector signals, and the BOLD signals are predicted by calcium within a given time period instead of at a single time point. To address this change, we treat the time series for each ROI as a 1D image and take different ROIs as image channels, so that we can take a 1D convolution at each layer of the network. There are two reasons for this design. First, the BOLD signals are local in time, meaning that they have a strong relation with the calcium signals that arise within the surrounding time period but a relatively weak relation with those at distant times. This can be modeled using the convolution operation since the receptive field of an output pixel is the surrounding input pixels defined by the size of the convolution kernels. Second, the BOLD signals are global in space. Since brain ROIs have functional connections with each other, the BOLD signals in one ROI may be affected by calcium signals from any other ROIs. It is reasonable to take ROIs as the channels of the images since the convolution operation is "fully-connected" across the input and output channel dimensions.

We still apply the WGAN-GP framework and the training object is given by

$$\min_G \max_D \mathcal{L}_{\text{WGAN-GP}}(G, D) + \mu_1 \mathcal{L}_{\text{L1}}(G) - \mu_2 \mathcal{L}_{\text{corr}}(G)$$

where $\mathcal{L}_{\text{WGAN-GP}}(G, D)$ and $\mathcal{L}_{\text{L1}}(G)$ are the WGAN-GP loss and L1 loss defined in (4.2) and (4.3). We also add a correlation loss

$$\mathcal{L}_{\text{corr}}(G) = \mathbb{E}_{x,y\sim p_{x,y}} \text{Corr}(y, G(x)) = \mathbb{E}_{x,y\sim p_{x,y}} \frac{\langle y, G(x)\rangle}{\|y\|\|G(x)\|}, \tag{4.4}$$

which encourages the correlation between predicted times series and the truth to be large. The architectures of the generator and discriminator are summarized in Table 4.4. The initial learning rate for both generator and discriminator is 0.0001, and it is divided by 2 after every 20 epochs. We take the gradient penalty coefficient $\lambda = 10$ as before. We take $\mu_1 = 1$ and $\mu_2 = 10$ from a grid search. In fact, a large L1 penalty μ_1 and small correlation loss penalty μ_2 would cause the generated BOLD time series close to zero, while larger μ_2 would help increase the variation of generated results.

Since the ROI data is available for 10 mice, we designs three disjoint test sets to test our model performance. The first test set includes the data for two random mice and contains 39 runs in total. The second one includes three random sessions from three remaining mice and contains 20 runs. The third one contains 21 runs that are randomly sampled from each of the remaining sessions. The rest of the data forms the training set. After the model is trained for 100 epochs, we find that the correlation between predicted BOLD times series and actual time series on the training data is 0.412. The correlations on the three test sets are 0.045, 0.045 and 0.043. This shows that model learns useful information from the data, although the correlations are relatively small compared with those on the training data. It is also surprising that the correlations are similar given that the three test sets have different levels of correspondence with the training set. This indicates that the model captures the general relationship between calcium and BOLD signals that is invariant for individual mice. We present the model predictions on training data and the first test data set in Fig. 4.7. We observe that on the training data, although the predictions do not exactly match the real time series, they always have a similar trend that is consistent with the correlation result. On the test data, most of the predicted BOLD time series still follow the real curves and look reasonable.

The functional connections between the ROIs can be characterized from the calcium or BOLD signals. A simple brain connectivity map is the connectivity matrix where the (i, j) entry is the correlation between calcium signals in the i-th and j-th ROIs if $i < j$, and the correlation between BOLD signals if $i > j$ (Lake et al., 2020). We can also evaluate our model by comparing the connectivity matrix computed from actual BOLD data and the connectivity matrix computed from the predicted BOLD time series. Fig. 4.8 gives the comparison between the real and predicted connectivity matrices on both training and test data. This result shows that the predictions retain the connectivity map, indicating the predictive power of our model.

4.5.1 Control Experiments

We observe that the correlation between the predicted BOLD time series and the actual time series on the test data are relatively small (0.045) compared to the correlation on the training data (0.412). It is possible that the trained model just randomly generates BOLD time series similar to those in the training data, regardless of the input of calcium time series. In this case, it becomes a generative model instead of translation model, and the connectivity matrix would also be preserved on the test data. We design two control experiments to test this hypothesis. In the first control experiment, for each pair of calcium and BOLD time series $(\mathbf{x}_i, \mathbf{y}_i)$ in the training data, we replace the calcium time series $\mathbf{x}_i = \{x_{i+t} | t = 0, 1, ..., 63\}$ with $\tilde{\mathbf{x}}_i = \{x_{j+t} | t = 0, 1, ..., 63\}$, where x_j is randomly picked from the 10-minute run that contains x_i. This effectively breaks up the relationship between calcium and BOLD on the time axis. In the second experiment, we shuffle the ROIs of the calcium time series $\mathbf{x}_i$ for each training pair $(\mathbf{x}_i, \mathbf{y}_i)$ so that the ROIs for input calcium signals are not aligned with the ROIs for target BOLD signals. We train the model under these two control settings using the same hyper-parameters and training/test data as before. Table 4.5 summarizes the correlations between predictive and real BOLD time series for these three experiments.

We observe that our translation model has significantly higher test correlation

Experiment	Training correlation	Test correlation
Normal setting	0.412	0.045
Scramble time	0.196	0.001
Shuffle ROIs	0.285	0.015

Table 4.5: Training and test correlations for normal translation model and two control experiments.

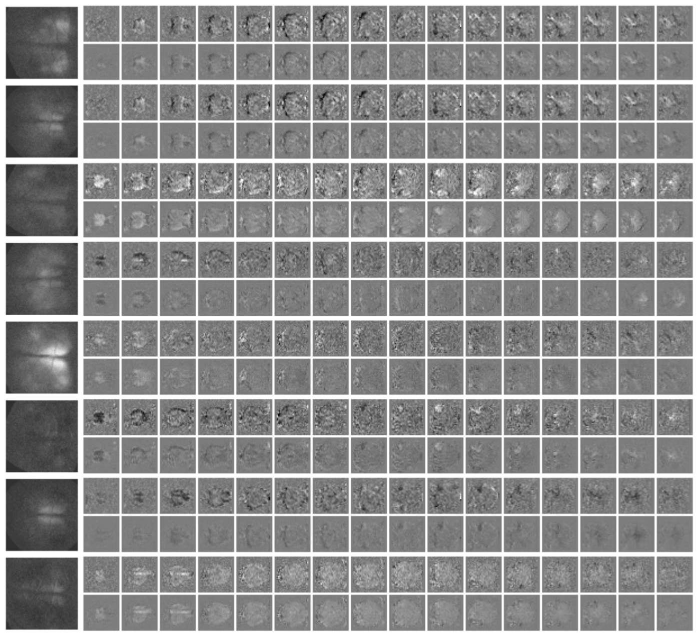
than the two control experiments, showing that the model indeed captures some of the relationship between calcium and BOLD signals. Notably, the test correlation for the second control experiment is still substantial. This is because the calcium signals are always consistent across a large number of ROIs, and some of the information is still preserved after shuffling the ROIs. We also examine the connectivity matrices for these two control experiments and find that they are similar to normal setting. This is still reasonable, since the presence of the discriminator network ensures the generated BOLD time series are similar to the actual time series, and the connectivity matrices should therefore be preserved as well.

4.6 Discussion

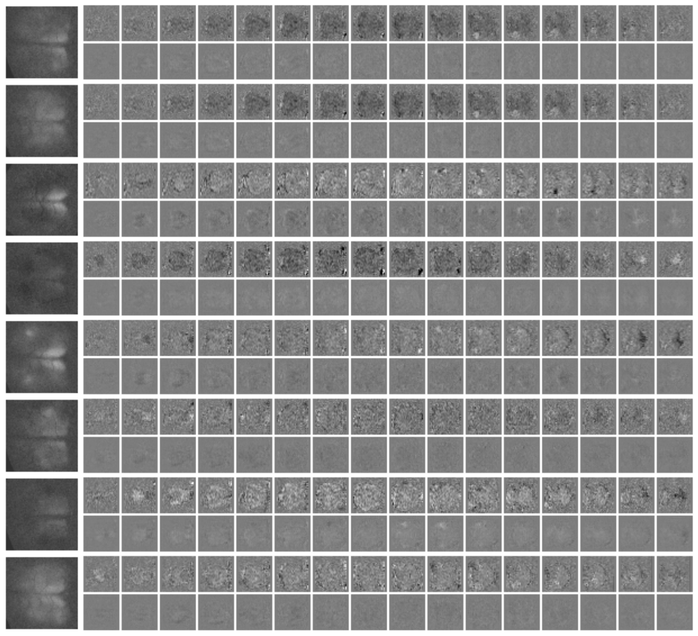
In this chapter, we present results of applying modern ANN models to the simultaneous mice fluorescence Ca^{2+} imaging and fMRI data sets. We start with generating calcium and BOLD images using WGAN-GPs, and show that the synthetic images have qualities that closely resemble the actual images. In following translation task, we try to map from calcium images to BOLD using conditional GANs. When BOLD images are not normalized, the brain anatomy is the dominant signal and variations of the BOLD images are quite small within each experiment session. In this case, the translation model can recover the anatomy signal of BOLD from the calcium images while the functional signals are overwhelmed. After normalizing the BOLD by subtracting the static anatomy, we find that the model is able to fit the training data to some extent but fails to generalize on test data — the predicted BOLD images always have small pixel values close to zero. There are several reasons that could lead to these results. First, the SNR of BOLD signals is relatively low compared to calcium given the nature of fMR imaging. The BOLD signals are measured in a low spatial and temporal resolution from outside of the brain, which introduce more noise to the collected BOLD images. Second, the calcium signals are consistent across the cortex, while the dynamics of BOLD signals are much more complicated and vary across layers of the brain. Third, the dynamics of BOLD signals may depend on factors other than calcium. It is especially hard to predict the BOLD signals at deeper layers of the brain from the calcium signals on the cortical surface. Finally, external factors such as the discrepancy in brain shape for different mice and the artifacts on the images brought by the device could also affect the model prediction power.

To address these challenges, we consider a simplified data set where the calcium and BOLD signals are averaged within the brain functional ROIs as determined by the Allen mouse brain atlas. This ROI data set reduces the high-dimensional images to low-dimensional signal vectors but still contains information of brain activity. Here we consider translating between the calcium and BOLD time series of length 64 seconds. We modify the conditional GAN model to adapt time series data and add a correlation loss to the training objective. We show that the predicted BOLD time series has high correlation with actual BOLD on training data, while the correlation is small but positive on the test set. Despite the small correlation, we observe that the predicted BOLD signals maintain the brain connectivity map across the ROIs by comparing the connectivity matrix between predicted and real BOLD signals. This result demonstrates the predictive power of our model and shows the merit of ANNs in the study of brain function.

We note that image translation between calcium and BOLD can be potentially improved in several ways in future studies. Given the uncertainty of BOLD signals at deep layers of the brain, it is reasonable to just focus on the BOLD images near the cortical surface and project those voxels to a 2D image, which serves as a new translation target. Moreover, more calcium images can be included in the model to predict a frame of BOLD, since a strong stimulus on the cortex may have long-term effects on BOLD. And a simple way to combine calcium images is stacking them along the channel dimension. The anatomical information could also be included in the model so that the model would be able to generate customized BOLD images for different mice. Finally, beyond an image translation model, a video generation or translation model (Tulyakov et al., 2018; Bansal et al., 2018) could also be applied to this data.

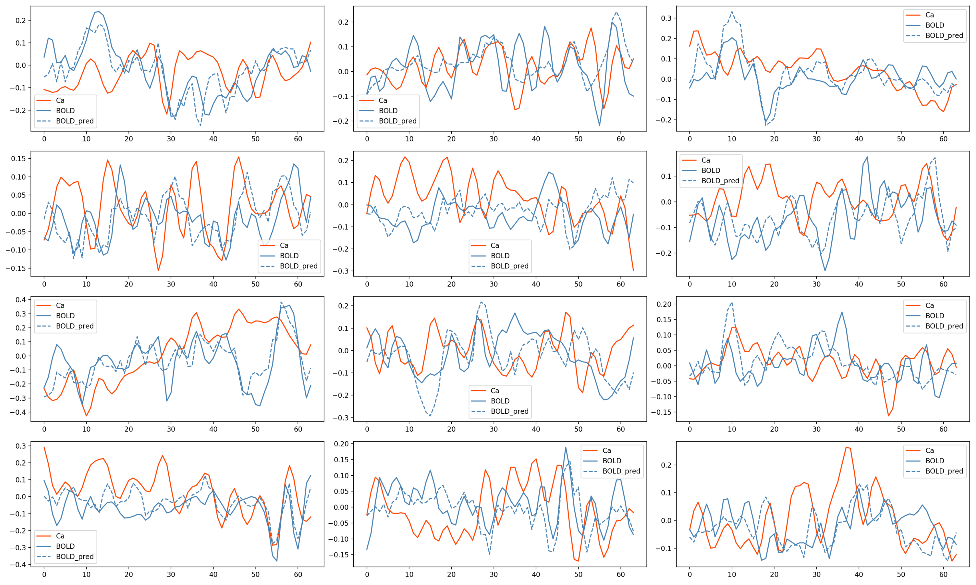


(a) Comparison between real and predicted BOLD images on training data.



(b) Comparison between real and predicted BOLD images on test data.

Figure 4.6: Predicting BOLD images from calcium images with `pix2pix` model, where BOLD images are normalized by subtracting the mean. For each of the eight rows in a figure, the input calcium image is on the left, the real 3D BOLD image is presented at the top sub-row on the right, and the predicted BOLD image is below the real one at the bottom sub-row.


(a) Comparison between real and predicted BOLD time series on training data.

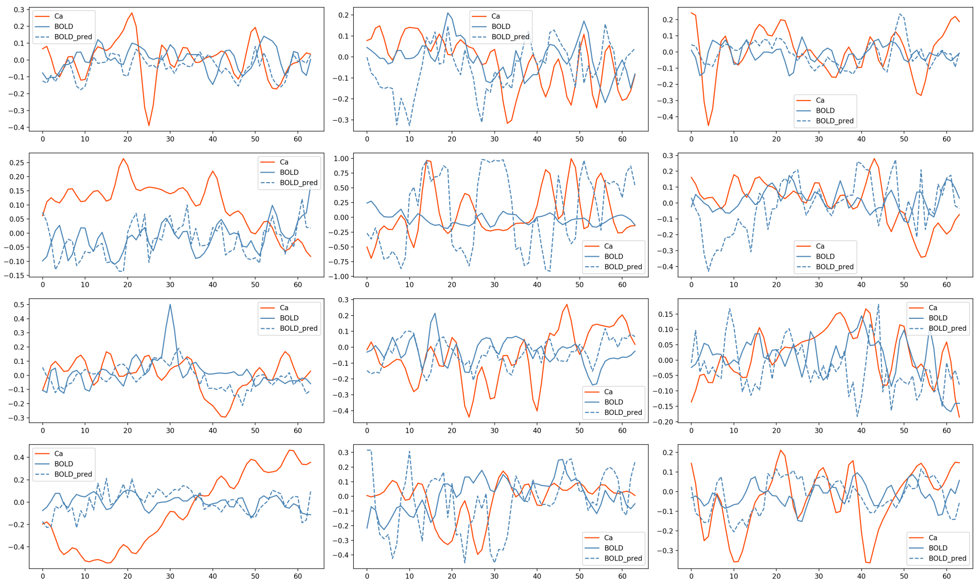
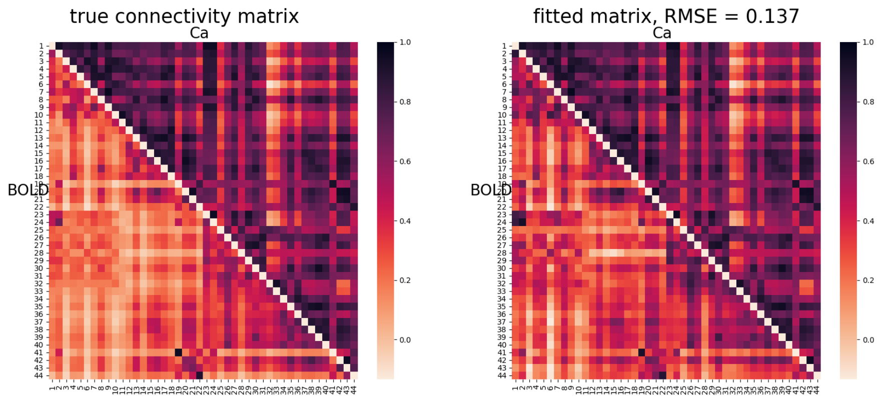
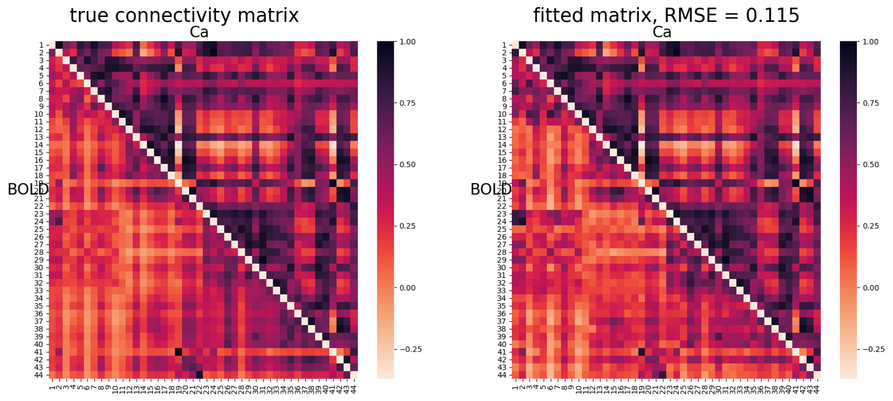

(b) Comparison between real and predicted BOLD time series on test data.

Figure 4.7: Predicting BOLD signals from calcium signals with `pix2pix` model on 14th ROI. Red curves, solid blue curves and dashed blue curves represent input calcium signals, real BOLD signals and predicted BOLD signals.



(a) Comparison of connectivity matrix on training data.



(b) Comparison of connectivity matrix on test data.

Figure 4.8: Comparison of connectivity matrix between one computed from real calcium & real BOLD signals (left) and one computed from real calcium & predicted BOLD signals (right). The title of the right figures gives the RMSE between two matrices.

Bibliography

Akrout, M., Wilson, C., Humphreys, P., Lillicrap, T., and Tweed, D. B. (2019). Deep learning without weight transport. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. *arXiv:1701.07875.*

Bansal, A., Ma, S., Ramanan, D., and Sheikh, Y. (2018). Recycle-gan: Unsupervised video retargeting. In *Proceedings of the European conference on computer vision (ECCV)*, pages 119–135.

Bartunov, S., Santoro, A., Richards, B., Marris, L., Hinton, G. E., and Lillicrap, T. (2018). Assessing the scalability of biologically-motivated deep learning algorithms and architectures. In *Advances in Neural Information Processing Systems*, pages 9368–9378.

Bellec, G., Scherr, F., Hajek, E., Salaj, D., Legenstein, R., and Maass, W. (2019). Biologically inspired alternatives to backpropagation through time for learning in recurrent neural nets.

Bora, A., Jalal, A., Price, E., and Dimakis, A. G. (2017). Compressed sensing using generative models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 537–546. JMLR. org.

Candes, E. J., Romberg, J. K., and Tao, T. (2006). Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics*, 59(8):1207–1223.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 6571–6583. Curran Associates, Inc.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., and Abbeel, P. (2016). Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180.

Davis, D., Drusvyatskiy, D., Kakade, S., and Lee, J. D. (2018). Stochastic subgradient method converges on tame functions. *Foundations of Computational Mathematics*, pages 1–36.

Deng, L. (2012). The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017). Density estimation using real NVP. arXiv:1605.08803.

Donoho, D. L. et al. (2006). Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306.

Du, S., Lee, J., Li, H., Wang, L., and Zhai, X. (2019). Gradient descent finds global minima of deep neural networks. In *International Conference on Machine Learning*, pages 1675–1685. PMLR.

Du, S. S., Lee, J. D., Li, H., Wang, L., and Zhai, X. (2018a). Gradient descent finds global minima of deep neural networks. *arXiv preprint arXiv:1811.03804*.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. (2018b). Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*.

Elman, J. L., Bates, E. A., Johnson, M. H., Annette Karmiloff-Smith, D. P., and Plunkett, K. (1996). *Rethinking Innateness: A connectionist perspective on development*. Cambridge MA: MIT Press.

Gao, C. and Lafferty, J. (2020). Model repair: Robust recovery of over-parameterized statistical models. *arXiv preprint arXiv:2005.09912*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein GANs. In *Advances in Neural Information Processing Systems*, pages 5767–5777.

Hand, P., Leong, O., and Voroninski, V. (2018). Phase retrieval under a generative prior. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Hand, P. and Voroninski, V. (2019). Global guarantees for enforcing deep generative priors by empirical risk. *IEEE Transactions on Information Theory*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hebb, D. O. (1961). Distinctive features of learning in the higher animal. In Delafresnaye, J. F., editor, *Brain Mechanisms and Learning*. London: Oxford University Press.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*.

Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4401–4410.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv:1412.6980*.

Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 10215–10224. Curran Associates, Inc.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014*.

Lake, E. M., Ge, X., Shen, X., Herman, P., Hyder, F., Cardin, J. A., Higley, M. J., Scheinost, D., Papademetris, X., Crair, M. C., et al. (2020). Simultaneous cortex-wide fluorescence ca 2+ imaging and whole-brain fmri. *Nature methods*, 17(12):1262–1271.

Laurent, B. and Massart, P. (2000). Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics*, pages 1302–1338.

LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

Li, Y. and Liang, Y. (2018). Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, pages 8157–8166.

Li, Y. and Yuan, Y. (2017). Convergence analysis of two-layer neural networks with relu activation. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 597–607. Curran Associates, Inc.

Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1):1–10.

Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., and Hinton, G. (2020). Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346.

Liu, M.-Y., Breuel, T., and Kautz, J. (2017). Unsupervised image-to-image translation networks. In *Advances in neural information processing systems*, pages 700–708.

Ma, Y., Shaik, M. A., Kozberg, M. G., Kim, S. H., Portes, J. P., Timerman, D., and Hillman, E. M. (2016). Resting-state hemodynamics are spatiotemporally coupled to synchronized and symmetric neural activity in excitatory neurons. *Proceedings of the National Academy of Sciences*, 113(52):E8463–E8471.

Medler, D. A. (1998). A brief history of connectionism. *Neural Computing Surveys*, 1:61–101.

Nøkland, A. (2016). Direct feedback alignment provides learning in deep neural networks. *arXiv preprint arXiv:1609.01596*.

Oja, E. (1982). A simplified neuron model as a principal component analyzer. *J. Mathematical Biology*, 15:267–273.

Paulsen, O. and Sejnowski, T. J. (2000). Natural patterns of activity and long-term synaptic plasticity. *Current Opinion in Neurobiology*, 10(2):172–179.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv:1511.06434*.

Rumelhart, D., McClelland, J., and the PDP Research Group (1986a). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 2: Psychologcal and Biological Models. Cambridge, Massachusetts: MIT Press.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986b). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Sejnowski, T. J. (1999). The book of Hebb. *Neuron*, 24:773–776.

Sejnowski, T. J. and Tesauro, G. (1989). The hebb rule for synaptic plasticity: Algorithms and implementations. In Byrne, J. H. and Berry, W. O., editors, *Neural Models of Plasticity*, pages 94–103.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Tulyakov, S., Liu, M.-Y., Yang, X., and Kautz, J. (2018). Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1526–1535.

Virmaux, A. and Scaman, K. (2018). Lipschitz regularity of deep neural networks: Analysis and efficient estimation. In *Advances in Neural Information Processing Systems*, pages 3835–3844.

Wager, S., Wang, S., and Liang, P. (2013). Dropout training as adaptive regularization. *arXiv preprint arXiv:1307.1493*.

Yamins, D. and DiCarlo, J. (2016). Using goal-driven deep learning models to understand sensory cortex. *Nature Neuroscience*, 19(3):356–365.

Yildirim, I., Wu, J., Kanwisher, N., and Tenenbaum, J. (2019). An integrative computational architecture for object-driven cortex. *J.B. Current Opinion in Neurobiology*.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017a). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232.

Zhu, J.-Y., Zhang, R., Pathak, D., Darrell, T., Efros, A. A., Wang, O., and Shechtman, E. (2017b). Multimodal image-to-image translation by enforcing bi-cycle consistency. In *Advances in neural information processing systems*, pages 465–476.