VIETNAM NATIONAL UNIVERSITY, HANOI
UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Nguyen Ngoc Hieu

AN ANALYSIS OF APPLYING NEURAL TANGENT KERNELS FOR DEEP EQUILIBRIUM MODELS

Major: Computer Science

HA NOI - 2022

VIETNAM NATIONAL UNIVERSITY, HANOI
UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Nguyen Ngoc Hieu

AN ANALYSIS OF APPLYING NEURAL TANGENT KERNELS FOR DEEP EQUILIBRIUM MODELS

Major: Computer Science

Supervisor : Dr. Ta Viet Cuong

Co-Supervisor : Dr. Hoang Thanh Tung

HA NOI - 2022

Supervisor: Dr. Ta Viet Cuong Nguyen Ngoc Hieu

Abstract

Deep Equilibrium model (DEQ) is a new class of models that directly computes the "infinite-depth" feature representation of a weight-tied network by solving a fixed point equation. These implicit-depth models have recently received much attention since they have been shown to achieve performance competitive with the traditional networks while using significantly less memory. Although these studies have experimentally proven its performances, the theoretical understanding of training DEQ models is limited. Our work studies the convergence of training DEQ models by leveraging the approach from the perspective of the Neural Tangent kernel (NTK). In this manuscript, we present our observation to explore the structure of the loss landscape of DEQ models. This observation suggests that DEQ models often have flat minimizers and raise questions about the trainability of DEQ models. Finally, we take a step toward answering these questions by using results from recent studies on the connection between the Neural tangent kernel and the training dynamic of neural networks. We prove that under suitable conditions, the convergence to the global optimum with linear rate is guaranteed.

Keyword: *Implicit Deep Learning, Deep Equilibrium Models, Gradient Descent, Learning Theory, Non-Convex Optimization, Neural Tangent Kernel*

Authorship

I hereby announce that the work in this thesis is original to me and has never been applied for a degree or diploma at this or any other higher education institution. Except where proper reference or acknowledgment is made, the study includes no materials previously reported or written by another author, to the best of my knowledge and belief.

Student

Nguyen Ngoc Hieu

Acknowledgements

I would like to thank a number of people for their help and support during the production of this thesis.

I want to express my sincere gratitude to my supervisor Dr. Ta Viet Cuong, as well as my co-supervisor Dr. Hoang Thanh Tung, for their guidance throughout this work. Their knowledge and experience were invaluable in formulating the research questions and approaches. This project would not have been possible without their support and encouragement.

I also want to express my gratitude to everyone from the FPT AI Residency program I've worked with and from whom I've learned so much.

Finally, I'd like to thank everyone else who has helped me persevere and prosper in the university.

Contents

List of Figures

Terminology

DEQ	Deep Equilibrium Models. 3
DNN	Deep Neural Network. 4
GD	Gradient Descent. 4
MDEQ	Multiscale Deep Equilibrium Models. 11, 25
monDEQ	Monotone Deep Equilibrium Models. 13
NTK	Neural Tangent Kernel. 4, 5
PL	Polyak-Lojasiewicz (condition). 18
SGD	Stochastic Gradient Descent. 15

Chapter 1

Introduction

"At the heart of modern deep learning methods is the notion of a layer."

Duvenaud et al. (2020)

1.1 Motivation

1.1.1 The depth of neural networks

Deep learning architectures are typically constructed by composing many parametric layers. This composition structure was inspired by the structure of the visual cortex in the brain and each layer corresponds to a different brain region (Figure 1.1). The depth can be defined as the length of the longest path from an input neuron to an output neuron.

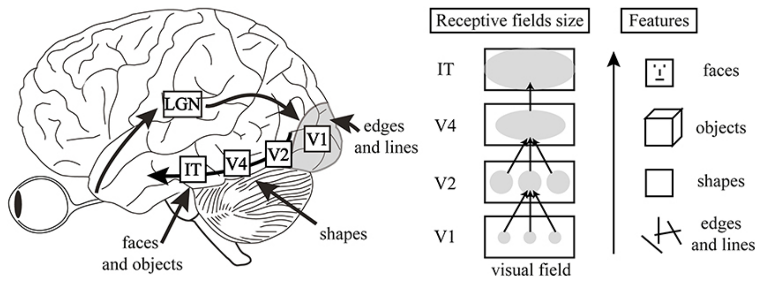


Figure 1.1: Features of the visual hierarchy. (Lindsay, 2021)

Even though it is theoretically possible to approximate any continuous function with a single hidden layer neural network (Hornik et al., 1989; Cybenko, 1989), the width might need to be exponentially large. In the expressivity view of neural networks, adding more layers can compensate for insufficient width (Hanin and Sellke, 2017; Hanin, 2019; Kidger and Lyons, 2020). The recent development of deep learning also empirically shows that increasing the depth of an NN can achieve impressive performance (He et al., 2015a). It leads to a long-held belief that the composition of several levels of layers would be the key to efficiently modeling complex relationships between variables and allows for more abstract features to be learned.

But is having better models as easy as having deeper/larger models? From the computation side, there are at least three drawbacks. A deeper network leads to more sequential processing and is harder to parallelize; making the computation of this network has higher latency and less suitable for applications that require fast responses. Another impediment is the memory limitations of available hardware. Deeper model often have more parameters and it is easy to hit hardware limitations as we try to scale our models given that current models often have hundreds of millions or even hundreds of billions of parameters. In distributed training, the obstacle comes from the communication overhead and directly proportional to the number of parameters in the model.

There are also questions about how we achieve "depth" in deep learning. Many recent works suggest that weight-tying models can perform as well as their fully connected counter pass in some particular tasks or even out-perform them in terms of systematic generalization (Schwarzschild et al., 2021; Csordás et al., 2021). These weight-tying models can increase their depth by simply doing more computation through a single layer and they might even have different depths with different inputs. This flexibility raises a natural question: "What if we have an infinite-depth weight-tied model?". This enables other approaches to the depth which taking the depth to its limits to result in continuous depth or infinite depth. This approach is called implicit layers, in contrast with explicit layers which are common in neural network architectures. Next, we will briefly introduce the term explicit and implicit layer.

1.1.2 Explicit vs Implicit layers

Explicit layer Let's consider a simple feedforward neural network built by stacking L layers, where L is the depth of the network. In the forward pass, each layer computes some transformation of the previous layer's output. To solve a particular task, we often create specialized layers that embed our inductive bias toward the task. For example, we typically have a convolution layer incorporating the local depend in image classification problems. Alternatively, we have recurrent layers (such as RNN, GRU, LSTM cells) that capture the temporal dependency for sequence data. These layers in modern deep learning are defined explicitly or specified by an exact sequence of operations to transform the input into the output. Let us take a convolution layer as an example. We precisely know how a feature map's pixel value is computed from the input and filters' weight as the convolution operation defines this computation,

$$(a \star b)_i = \sum_{j \in G} a_j b_{j^{-1}} i, i \in G$$

and typically followed by element-wise nonlinearities like ReLU, with additional operations like normalization or dropout. Many convolution layers can be connected in multiple different ways to form things like residual layers. These kinds of layers that provide a concrete computation graph for how we compute the output from the input are called explicit layers.

Implicit layer. In contrast, a new class of deep learning models is based on implicit prediction rules called implicit layers. Instead of having a concrete computation graph, these layers define a layer in terms of satisfying some joint conditions of the input and output i.e. *finding* z such that it satisfies some constraint e.g, z is required to be a root of equation $g(x, z) = 0$. In practice, this formalism can capture algebraic equations and fixed points such as $z = f(z, x)$, leading to deep equilibrium models (Bai et al., 2019a); or differential equations $\frac{dz(t)}{dt} = f(z(t), t)$, leading to Neural ODEs (NODEs) (Chen et al., 2018); or the optimality conditions of optimization problems, leading to differentiable optimization $z(x) = \underset{z \in \mathcal{C}(x)}{\arg\min} f(z, x)$ approaches (Amos and Kolter, 2017).

Recent progress on implicit networks has motivated this novel class of networks to the forefront of deep learning research. There is a wide array of applications that have been addressed using implicit layers (Duvenaud et al., 2020). In traditional problems in machine learning, some implicit models achieve performance on par with state-of-the-art Transformer models (at the same parameter count) for language modeling, and on par with state-of-the-art computer vision architectures on tasks such as classification and semantic segmentation (Bai et al., 2019a, 2020). Moreover, they can solve structured convex problems or smoothed relaxations of combinatorial optimization problems. These include many practical problems such as graph cuts, satisfiability, and many others (Duvenaud et al., 2020; Amos and Kolter, 2017; Wang et al., 2019; Djolonga and Krause, 2017). With the ability to integrate differential equations as layers in deep networks, these models have been used in integrating continuous time observations or approximating continuous versions of traditional residual networks (Rubanova et al., 2019; Kidger et al., 2020; Brouwer et al., 2019). We can also create architectures using implicit layers for efficient representation of smooth densities, for use in generative modeling and beyond (Song et al., 2021; Yang et al., 2019; Mathieu and Nickel, 2020; Lou et al., 2020; Falorsi and Forré, 2020).

This work will focus on deep equilibrium models (DEQs) Bai et al. (2019a), a class of implicit layers that define the output as a fixed point of a state transition mapping and can be viewed as an infinitely deep network. The DEQ shares some similarities with the Recurrent Backpropagation (RBP) algorithm (ALMEIDA, 1987; Pineda, 1987), which is a very memory-efficient algorithm to obtain gradients of a particular type of recurrent neural network with convergent hidden state dynamics.

1.1.3 Optimization and Generalization

Optimization. The implicit layers are a new class of layers to build neural networks, and as we will see, back-propagation through these layers can be computed efficiently. But why do we expect that the training process will converge? Neural network-based machine learning is well-known for being somewhat of a "black magic." Its success relies on many tricks; for example, parameter tuning can be quite an art. The conventional belief was that the loss landscape of neural networks with more than two layers is nonconvex, highly nonlinear, and high-dimensional. So it was not trivial at all to discover that deep neural networks (DNNs) can be trained. Understanding why first-order algorithms like gradient descent (GD) based methods can often find a global optimum in optimizing the non-convex loss surface of neural networks is one of the critical problems in deep learning theory. This work aims to take a step toward answering the above question for DEQs, a class of architecture that differs from those commonly used in the related literature.

Generalization. In terms of generalization performance, deep learning often works in the regime where the number of trainable parameters is larger than the size of the training dataset a.k.a over-parameterized regime. Conventional prescriptions from learning theory would suggest that one should expect overfitting in this regime. Unlike the under-parameterized regime where the loss function largely determines the solution. It is often the case that, in the over-parameterized regime there is an infinite number of global minimizers of the loss function. When many global minima exist, a difference in the gradient dynamics can lead to a substantial disparity in the learned models. Two different gradient dynamics can end up with very different global minima which have different performances for generalization (Kawaguchi et al., 2017). So the final solution which is picked depends on the details of the training algorithm. But there is no guarantee that such an algorithm-dependent solution can generalize to unseen data. This is called implicit bias which is defined as the inductive bias induced implicitly through gradient dynamics and is the subject of an active research area in machine learning theory.

Recent development In an effort to shed light on what makes NNs perform so well under versions of gradient descent, there is a large body of work dedicated to understanding this problem for traditional neural networks. There are 2 lines of research. The first one is to describe geometric landscapes of the objective function. Recent advance in the ”first-order” algorithm showed that if the landscape satisfies 2 conditions: (1) all local minima are global and (2) all saddle points are strict (i.e, there exist a negative curvature), then first-order methods such as GD/SGD and its variance can escape all saddle points and find a global minimum (Ge et al., 2015; Lee et al., 2016; Jin et al., 2017). Unfortunately, these expected properties do not hold even for simple non-linear 2-layer neural networks (Yun et al., 2018) or deep linear neural networks (Kawaguchi, 2016). One recent idea is to analyze NNs through the trajectory-based approach. The theory emerges when the network width tends to infinity. With a set proper scaling factor on the activation, this over-parameterization regime is called Neural Tangent Kernel (NTK) regime (Jacot et al., 2018). In this NTK regime, instead of analyzing the trajectory of the parameter, we keep track of the trajectory of predictions of neural networks, and via their kernel approximation, it can be shown that the gradient descent algorithm explores only a small region. This result can be used to prove the global polynomial time convergence of gradient descent on non-linear neural networks for minimizing the objective function.

1.2 Contributions and thesis overview

Despite the remarkable performances of DEQs, our theoretical understanding of its properties is yet limited. This thesis try to take a step forward into theoretical understanding of training DEQ models. In particular, we aim to answer a key question: "Why gradient descent methods can often find a global optimum in training DEQ models?". Answering this question not only affects our understanding of training DEQs but also sheds light on understanding the generalization properties of these models. Many works have studied this problem for explicit deep neural networks, however, it is not straightforward to apply these results to DEQ models, since their output is implicitly defined and might not be well-pose. Kawaguchi (2021) study gradient dynamics of deep equilibrium linear models but their result cannot be extended to nonlinear models. Ling et al. (2022); Gao and Gao (2022) study convergence of gradient descent for ReLU DEQ model but only in the contraction regime. In this work, we study the convergence of training monotone DEQ models with the ReLU activation function, which is a richer class of models.
The main contribution of this thesis is summarized as follows.

1. We analyze the gradient flow dynamic of DEQs with the ReLU activation function and show that despite the "infinite" depth, non-smooth, and non-convexity, the linear convergence rate to a global optimum is guaranteed if the network satisfies conditions at initialization.

2. We show gradient descent with fixed step size converges to a global minimum of DEQ networks at a linear rate as long as the same assumptions are satisfied and the step size is small enough.

The rest of this thesis is organized as follows.

- Chapter 2 is dedicated to describe preliminary knowledge. The first three sections in this chapter lay the foundations for understanding DEQ model. We provide a background of deep equilibrium models which include its definition of output, the forward pass, and the backward pass. The rest of this chapter develops the theory of training neural networks and presents a powerful theoretical tool called Neural Tangent Kernel (NTK). This theory lays mathematically rigorous foundations for our results.

- Chapter 3 present our work on analyzing optimization of DEQ models by using NTK. It includes our observation of the loss landscape and a mathematical framework based on the training dynamic and NTK.

- Chapter 4 present our experiments with DEQs.

- Chapter 5 conclude the thesis's work and present our interest in future works.

1.3 Notation

Table 1.1: Table of Notation

	Symbol/E.g	**Description**
	Variables	
Bold lowercase letters	$\mathbf{v}$	vectors
Capital letters	W	matrices
	Operators	
	$\langle \cdot, \cdot \rangle$	Inner product
	$\|\cdot\|$	Vector norm or operator norm
	$det(\cdot)$	Determinant
	$\rho(\cdot)$	Spectral radius
	$\lambda_{\max}, \lambda_{\min}(\cdot)$	The maximal/minimal eigenvalue value
	$\sigma_{\max}, \sigma_{\min}(\cdot)$	The maximal/minimal singular value
	$\otimes$	Kronecker product
	$\oplus$	Kronecker sum
	$\circ$	Hardarmat product
	$[n]$	The set $\{1, 2, \dots, n\}$

Chapter 2

Background

In order to understand the training dynamic of DEQ models, we first need to understand (1) how DEQ models work, and (2) training neural network dynamic.

The first 3 section in this chapter are dedicated for (1). We will start off by definition of DEQ layers which includes its definition of output, the forward pass, and the backward pass. Some instantiations will also be discussed, highlighting the benefits of using DEQ networks to solve many challenging tasks such as language modeling, image classification, semantic segmentation, and inverse problems in imaging. We also highlight some problems of using DEQ model and present some recent works that address these problems of DEQ.

The last three are dedicated for (2). They develops the theory of training neural network and presents a powerful theoritical tool, Neural Tangent Kernel, proposed by Jacot et al. (2018). This theory lays mathematically rigorous foundations for our results.

2.1 Definition of a DEQ layer

2.1.1 The output

Let $f_\theta : \mathbb{R}^{d_z} \times \mathbb{R}^{d_x} \mapsto \mathbb{R}^{d_z}$ is a parametrized function, which takes a parameter vector $\theta \in \mathbb{R}^{d_\theta}$. The function f_θ can be a layer (usually a shallow block; e.g self-attention, a recurrent cell or a resnet block). The output $\mathbf{z}^\star$ of a DEQ layer for input x is characterized by the fixed point of the sysem:

$$\mathbf{z}^\star = f_\theta\left(\mathbf{z}^\star, \mathbf{x}\right) \tag{2.1}$$

where $\mathbf{z}^\star$ is a equilibrium point of the evolutionary operator f_θ.

The benefits of using fixed point DEQ explicitly define its output as a fixed point of a dynamical system. There are some benifits when using fixed point as the output of the layer.

1. **Modeling** There are empirical evidence suggests that the fixed point of DEQ may have a rich feature representation. We present these evidence as the motivation of DEQ

models in appendix B. Moreover, since DEQs find fixed points by design, they are well suited to problems in areas as diverse as game theory and inverse problems where the output of interest may naturally be characterized as the fixed point to an operator parameterized by the input data $\mathbf{x}$. This is show in recent works (Heaton et al., 2021; Gilton et al., 2021).

2. **Expressiveness** The representation from DEQ layer might be powerful. With DEQ, we have gone from a deep network to literally trying to compute every thing with a single implicit layer. But are DEQ networks as expressive as normal network? Bai et al. (2019a) has showed that "Any deep network (of any depth, with any connectivity), can be represented as a single layer DEQ model." Furthermore, this does not involve the kind of exponential parameter blowup common in single layer universal function approximation theorems: with the same number of parameters, a single layer DEQ can represent any network.

3. **"Infinite depth" with constant memory training** An important benefit of DEQ is its extreme memory efficiency. A typical deep network will take the form of stacking of many layers or a combination of many different functions. And the calculation of $\nabla_\theta f_\theta(x)$ is done by building a computation graph showing the calculations needed to get the output from the input and using a chain rule on this graph. This process requires memory to store intermediate value in forward pass on this computational graphs. A DEQ only needs to store $\mathbf{z}^*$ (the equilibrium point), $\mathbf{x}$ (input-related, layer-independent variables), and the parameter θ for the backward pass.

2.1.2 Forward pass

Denote $f(\mathbf{z}, \mathbf{u}) = f_\theta(\mathbf{z}; \mathbf{x})$ where $\mathbf{u} \in \mathbb{R}^{d_x + d_\theta}$ is the union of the input $\mathbf{x} \in \mathbb{R}^{d_x}$ and parameters $\theta \in \mathbb{R}^{d_\theta}$, i.e, $\mathbf{u}^T = [\mathbf{x}^T, \theta^T]$. We can construct a function g of $\mathbf{u}$ and $\mathbf{z}$ as follows,

$$g(\mathbf{z}, \mathbf{u}) = f(\mathbf{z}, \mathbf{u}) - \mathbf{z}. \tag{2.2}$$

At the fixed point, we have $g(\mathbf{z}^*, \mathbf{u}) = 0$. So the forward pass of DEQ is basically finding the solution of 2.2.

Since DEQ explicitly uses equilibrium as a replacement for depth in general networks, it can separate the solution procedure of the layer from the definition of the layer itself. That mean the forward evaluation of DEQs could be any procedure that solves 2.2.

The simplest method is naive forward iteration or fixed-point iteration, where we iterate $\mathbf{z}^{[i+1]} = g(\mathbf{z}^{[i]}, \mathbf{x})$ until $\mathbf{z}^{[i+1]}$ stays sufficiently close to $\mathbf{z}^{[i]}$. Similar results can be obtain by infinitely iterating weight-tied input-injected layer. Whether this method succeeds depends on how we initialize it and on properties of f (for example, f is a contraction mapping on $\mathbf{z}$). And even if it does succeed, it usually needs many iterations before convergence is reached. Newton's method is also a inefficient way to doing this because it requires forming an enormous Jacobian which can't be done in practice.

These simple iteration method forgets about all the information gained from the previous iterations result in slow convergence. There are accelerated fixed point methods that take the output of previous iterations into account and combine them in a much better way. In practice, DEQ models usually use these accelerated fixed point methods. Bai et al. (2019a, 2020); used Broyden's method (Broyden, 1965), the memory consumption of which grows linearly with the number of iterations since all low-rank updates are stored. Other recent work (Duvenaud et al., 2020; Gilton et al., 2021) shifted to Anderson acceleration (AA) (Anderson, 1965), a lightweight solver that is provably equivalent to a multi-secant quasi-Newton method (Fang and Saad, 2009).

Recently, Bai et al. (2022) proposes to speed up inference of Deep Equilibrium Models (DEQs) by replacing the classic fixed-point solvers (Broyden or Anderson Acceleration) with a learned extension of Anderson Acceleration. Their approach operates on a pre-trained DEQ and trains a small neural network to propose an initialization and update scheme based on ground truth fixed points. Their experiments show that these modifications substantially improve the speed/accuracy trade-off across diverse large-scale tasks while adding almost no overhead to training.

2.1.3 Backward pass

In backward pass, traditional back propagation will not work effectively for DEQs since it requires to store intermediate hidden state of the forward propagation which could entail hundreds or thousands of iterations, requiring ever growing memory to store computational graphs. Naively different thought the iteration of a solver also has other disadvantage like doing more floating point operation and as the number of unrolling steps increases, the numerical error accumulates which may render the algorithm useless in some applications e.g., gradient based hyper parameter optimization Maclaurin et al. (2015).

Bai et al. (2019a) provide an alternative procedure which is based on RBP which is a algorithm to train recurrent neural networks which has a convergent hidden state dynamics, and benefit from the same advantages as RBP: memory efficient.

Assuming some proper conditions on g in equation 2.2

1. g is continuous differentiable, and
2. $\nabla_{\mathbf{z}} g(\mathbf{z}^*, \mathbf{u}) = J_g(\mathbf{z}^*) = J_f(\mathbf{z}^*) - I$ is nonsingular

where $J_f(\mathbf{z}^*) = \frac{\partial f(\mathbf{z}^*, \mathbf{u})}{\partial \mathbf{z}}$ is the Jacobian matrix of f evaluated at $\mathbf{z}^*$. Implicit function theorem guarantees the existence and uniqueness of an implicit function s such that $\mathbf{z}^* = s(\mathbf{u})$. Although we do not know the analytic expression of the function s, we can still compute its gradient at the fixed point

$$\frac{\partial \mathbf{z}^*}{\partial \mathbf{u}} = -J_g(\mathbf{z}^*)^{-1} \frac{\partial g(\mathbf{z}^*, \mathbf{u})}{\partial \mathbf{u}} = \left(I - J_f(\mathbf{z}^*)\right)^{-1} \frac{\partial f(\mathbf{z}^*, \mathbf{u})}{\partial \mathbf{u}} \tag{2.3}$$

Typically, there is a output function $\hat{y} = p_\psi(\mathbf{z}^*)$, which is parameterize by ψ, to obtain a prediction and a loss function $\ell(p_\psi(\mathbf{z}^*), y)$ measures the closeness between ground truth y

and predicted output $\hat{y}$. Based on Eq. 2.3, we now turn our attention towards computing the gradient of the loss ℓ w.r.t. the parameters θ and ψ of the network. By using the total derivative and the chain rule, we have

$$\begin{aligned} \frac{\partial \ell}{\partial \psi} &= \frac{\partial \ell}{\partial \hat{y}} \frac{\partial p_{\psi}(\mathbf{z}^*)}{\partial \psi} \\ \frac{\partial \ell}{\partial \theta} &= \underbrace{\frac{\partial \ell}{\partial \mathbf{z}^*} \left(I - J_f(\mathbf{z}^*)\right)^{-1}}_{\mathbf{v}^{\top}} \frac{\partial f(\mathbf{h}^*; \mathbf{u})}{\partial \theta} \end{aligned} \tag{2.4}$$

Since given the output $\mathbf{z}^*$ which is the steady hidden state of f_θ, training the post-processing part p_ψ is simply back propagation, we omit the parameters of p_ψ and focus on the gradient of the loss function ℓ w.r.t. the parameter of f_θ which is the most crucial component. To bypass the explicit calculation of the inverse $\left(I - J_f(\mathbf{z}^*)\right)^{-1}$ which is costly to compute for large scale, the original RBP algorithm Pineda (1987), ALMEIDA (1987) introduces an auxiliary variable $\mathbf{v}^\top$ in term of Eq 2.4 such that and compute it by solving the following linear fixed-point system which can be solved using the same solver as the forward pass.

$$\mathbf{v}^\top = \mathbf{v}^\top J_f(\mathbf{z}^*) + \left(\frac{\partial \ell}{\partial \mathbf{z}^*}\right)$$

Note that the most expensive operation in this algorithm is the vector-matrix product $\mathbf{v}^\top J_f(\mathbf{z}^*)$, which is the same operator as back-propagation and can be efficiently computed by using auto differentiation tools.

The most important message from Eq. 2.4 is that the backward pass can be computed with merely the knowledge of $\mathbf{z}^\star$, irrespective of how it is found. This enable the backward procedure to requires constant memory since we do not store any intermediate activation during the forward pass and need to store only $\theta, \mathbf{z}^*$ and $\mathbf{x}$ for the backward pass. Assuming no knowledge of the black-box solver for the equilibrium point in the forward pass, one can implicitly differentiate through $\mathbf{z}^\star$, and produce gradients with respect to the model parameters θ.

2.2 Application of DEQ models

In practice, DEQs are competitive with the state of the art in deep learning for similar size and similary train architectures.

Language modeling task For the language modeling task, Bai et al. (2019a) trained a DEQ variant of a transformer model (Vaswani et al., 2017) as well as a DEQ variant of a Trellis network (Bai et al., 2019b) to model a language modeling task on WikiText-103, a standard reasonably large scale dataset for Language modeling. The DEQ models typically get better performances for the same number of parameters, saving up to 88% memory, even outperforming gradient checkpointing for trellis network on large scale datasets, with almost no accuracy drop.

Vision task In vision tasks, depth plays two roles. In one sense, the depth increases the network's capacity, but it typically also involves sub-sampling data or strided convolutions that downsample data. Depth helps create representations of the image at multiple difference feature resolutions. Multiscale DEQ (MDEQ) presented an idea that extends these DEQ models to a multi-scale setting. The idea is to simultaneously represent multiple spatial scales of features in the hidden unit, feed each through a residual block, and mix them all by upsampling and downsampling. Then treat this whole thing as the fixed point equation. This model has feature maps for both high resolution and low resolution, which lets us use the same model for a task like image classification as for a task like image segmentation. By incorporating structure within a single "layer" (cell), we can create a very powerful model that has a lot of structure within it and can solve many challenging tasks.

MDEQs on ImageNet for a similar size of models and similar training techniques are competitive with other architectures like Resnets, HRNet, DenseNet, e.t.c. Applying MDEQs to semantic segmentation on the Citiscapes dataset also gets competitive performances with state-of-the-art while using the same model for classification.

Inverse Problems in Imaging Recent efforts on solving inverse problems in imaging via deep neural networks use architectures inspired by a fixed number of iterations of an optimization method. The number of iterations is typically relatively small due to difficulties in training networks corresponding to more iterations; the resulting solvers cannot be run for more iterations at test time without significant errors. Gilton et al. (2021) uses an approach to find a fixed point of an operator that corresponds to an infinite number of iterations, consistently improving reconstruction accuracy above state-of-the-art alternatives. This paper illustrates the non-trivial quantitative benefits of using DEQ networks to solve linear inverse imaging problems.

2.3 The well-posedness problem

While conventional explicit layers and some the other implicit layers such as Neural ODEs guarantee a unique solution, obtaining convergence in DEQs requires careful initialization and regularization, which has proven difficult in practice. Moreover, existence of a unique fixed point is not guaranteed to exist or be unique, making the output of the models potentially ill-defined. So DEQs are reliant on meticulous architectural designs. We highlight some of these properties in our experiment (section 4.1). Bai et al. (2021) has systematic discuss the brittleness to architectural choices of DEQ by ablative studies on the use of layer normalization (LN) or weight normalization (WN) in the DEQ-Transformer model on the large-scale WikiText-103 language modeling task. The need to have a relative stable DEQ in order to trained it via the implicit function theorem calls for more carefull attention in designing the layer f_θ. How should we choose the layer f_θ to guarentee the existence of equilibrium point and also uniqueness or stability?

2.3.1 Contraction function

Recall that in order to apply the implicit function theorem, f has to satisfy two assumptions: (1) f is continuously differentiable, and (2) $I - J_f(\mathbf{z}^*)$ is invertible. Condition (1) requires the derivative of f to be continuous, a condition satisfied by many RNNs cell, like LSTM and GRU (Cho et al. (2014),). Condition (2) is equivalent to requiring the determinant of $I - J_f(\mathbf{z}^*)$ to be nonzero, i.e., $\det(I - J_f(\mathbf{z}^*)) \neq 0$. One sufficient but not necessary condition to ensure this is to force f to be a contraction map, as in Scarselli et al. (2009).

Definition 2.3.1 (Contraction mapping). *F is a contraction map on Banach space B, i.e., a complete normed vector space, iff, $\forall \mathbf{h}_1, \mathbf{h}_2 \in B, \|F(\mathbf{h}_1) - F(\mathbf{h}_2)\| \leq \mu \|\mathbf{h}_1 - \mathbf{h}_2\|$ where $0 \leq \mu < 1$.*

Banach fixed point theorem guarantees the uniqueness of the fixed point of the contraction map F in B. Based on the first order Taylor approximation, for $\mathbf{z}$ sufficiently close to $\mathbf{z}^*$, $f(\mathbf{z}, \mathbf{u}) = f(\mathbf{z}^*, \mathbf{u}) + J_f(\mathbf{z}^*)(\mathbf{z} - \mathbf{z}^*)$, we have

$$\frac{\|f(\mathbf{z}, \mathbf{u}) - f(\mathbf{z}^*, \mathbf{u})\|}{\|\mathbf{z} - \mathbf{z}^*\|} = \frac{\left\|J_f(\mathbf{z}^*)\,(\mathbf{z} - \mathbf{z}^*)\right\|}{\|\mathbf{z} - \mathbf{z}^*\|}. \tag{2.5}$$

If we use L_2 vector norm, then the induced matrix norm, a.k.a., operator norm, is,

$$\left\|J_f(\mathbf{z}^*)\right\| = \sup\left\{\frac{\left\|J_f(\mathbf{z}^*)\right\|}{\|\mathbf{z}\|} : \forall \mathbf{z} \neq 0\right\} = \rho\left(J_f(\mathbf{z}^*)\right) \tag{2.6}$$

where ρ is the spectral radius (i.e the largest singular value).

$$\rho\left(J_f(\mathbf{z}^\star)\right) = \rho\left(J_f(\mathbf{z}^\star)^\top\right) = \max\left(|\lambda_1|, \ldots, |\lambda_d|\right)$$

Therefore, relying on the contraction map definition, we have

$$\left\|J_f(\mathbf{z}^*)\right\| \leq \mu < 1, \tag{2.7}$$

Moreover, since the minimum singular value of $I - J_f(\mathbf{z}^*)$ is $1 - \rho(J_f(\mathbf{z}^*))$, we have

$$\begin{aligned} \left|\det\left(I - J_f(\mathbf{z}^*)\right)\right| &= \prod_i \left|\sigma_i\left(I - J_f(\mathbf{z}^*)\right)\right| \\ &\geq \left[1 - \rho\left(J_f(\mathbf{z}^*)\right)\right]^d > 0. \end{aligned} \tag{2.8}$$

Thus our second condition holds following Eq 2.8 and guarantees that there exists a unique input dependent fixed point $\mathbf{z}^*$ for every input $\mathbf{x}$. Moreover, the naive fixed point iteration could converge uniquely.

However, the contraction mapping property is rather a strong assumption since even if f is not a contraction mapping, a fixed point can still exist and the much stronger root solvers are still able to solve for them, as noted by Bai et al. (2021) Liao et al. (2018). In practice, we don't always, and probably shouldn't, require such a strong contractivity on the dynamical system, which might significantly limit the representational capacity of the model. It can be weakened to an assumption of local contraction around the fixed point which can be achieved by regularizing the local Jacobian $J_f(\mathbf{h}^*)$ of a general neural network. Bai et al. (2021)

2.3.2 Monotone DEQ

We can also guarentee the existence of a unique fixed point by reparametrize the network. Monotone DEQs (monDEQs) Winston and Kolter (2020) studies a new way of parameterizing the DEQ by exploiting provably convergent layers via monotone operator splitting theories. It admits a Lipschitz bound during training via unconstrained optimization. First, they notice a connection between the fixed point of DEQ equation and solution of operator splitting problem which has the form as 1.

Problem 1. Finding a zero of the operator splitting problem $0 \in (F+G)(\mathbf{z}^\star)$ with the operators

$$F(\mathbf{z}) = (I-W)(\mathbf{z}) - (U\mathbf{x}+\mathbf{b}), \quad G = \partial f$$

and $\sigma(\cdot) = \text{prox}_f^1(\cdot)$ for some convex closed proper (CCP) function f, where prox $^\alpha_f$ denotes the proximal operator

$$\text{prox}_f^\alpha(x) \equiv \underset{z}{\text{argmin}} \frac{1}{2}\|\mathbf{x}-\mathbf{z}\|_2^2 + \alpha f(\mathbf{z}).$$

Operator splitting approaches refer to methods to find a zero in a sum of operators (such as problem 1). There are many such operator splitting methods, but the two they use mainly are 1) forward-backward splitting and 2) Peaceman-Rachford splitting. In order to guarantee the convergence to a unique solution, these methods require maximal monotonicity which is equivalent to a convex closed proper (CCP) function.

Lemma 2.3.1. *A subdifferentiable operator ∂f is maximal monotone iff f is a convex closed proper (CCP) function.*

Using lemma 2.3.1, we have the operator G is maximal. For the operator F, we leverage following lemma

Lemma 2.3.2. *A linear operator $F(\mathbf{x}) = W\mathbf{x} + h$ for $W \in \mathbb{R}^{n\times n}$ and $h \in \mathbb{R}^n$ is (maximal) monotone if and only if $W + W^T \succeq 0$ and strongly monotone if $W + W^T \succeq mI$.*

Both methods will converge linearly to an $\mathbf{z}$ that is a zero of the operator sum under certain conditions:

- A sufficient condition for forward-backward to converge is that F be strongly monotone with parameter m and Lipschitz with constant L and $\alpha < \frac{2m}{L^2}$.
- For Peaceman-Rachford, the method will converge for any choice of α for strongly monotone F, though the convergence speed will often vary substantially based upon α.

So we can establish the existence and uniqueness of the solution via the criterion that F is strongly monotone. Using lemma 2.3.2, F is strongly monotone if $I - (W + W^T)/2 \succeq mI \Leftrightarrow I - W \succeq mI$. Then they reparameterize W in a manner such that it is guarantee to have a unique solution based on following propsition

Proposition 1 (Propsition 1 in Winston and Kolter (2020))**.** *We have $I - W \succeq mI$ if and only if there exist $A, B \in \mathbb{R}^{n \times n}$ such that*

$$W = (1-m)I - A^\top A + B - B^\top$$

.

2.4 Learning as optimization

At an abstract level, learning is optimizing over data. In other words, the network learns the input-output relationship in the data through optimization. For example, in supervised learning, the parameters of a neural network are optimized so that given input in the data, the network output a corresponding (or approximate) target in the data. Assume the inputs $\mathbf{x}$ and labels y are jointly draw from the data generating distribution, i.e $(\mathbf{x}, y) \sim p_{\text{data}}$ We are given a training set of size N, $\{(\mathbf{x}_i, y_i\}_{i=1}^N$. and the predictions are made using a family of models $f(\theta, \mathbf{x})$ (e.g a network architecture) parameterized by θ. One aim is to "learn" θ that fits the training data, i.e.,

$$f(\theta; \mathbf{x}_i) \approx y_i, i = 1, 2, \ldots, N.$$

We determines how fit we are by a loss function $L(f(\theta; \mathbf{x}_i), y)$. Mathematically, learning is equivalent to finding a model with parameter θ^* which achieves small risk, or generalization loss

$$\mathcal{R}(\theta) = \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}}\left[L(f(\theta, \mathbf{x}), y)\right]$$

We do this by minimizing a cost function corresponding to the empirical risk over the training set

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} L(f(\theta; \mathbf{x}_i), y_i) \tag{2.9}$$

We can write this as a basic unconstrained optimization problem

$$\arg\min_{\theta \in \mathbb{R}^p} \mathcal{L}(\theta) \tag{2.10}$$

Gradient Descent A common optimization algorithm to solve this problem is gradient descent (GD). The general idea of gradient descent is based on initializing θ at random and performing the sequential updates where each iteration updates the weights opposite the gradient direction:

$$\theta_{k+1} \leftarrow \theta_k - \eta \nabla \mathcal{L}(\theta_k) \tag{2.11}$$

The convergence depends on η and the structural properties of $\mathcal{L}$ itself. GD is a batch method which mean it use the full training set to compute the next update to parameters at each iteration. And even though it is straightforward to get GD working, it also tends to converge very well to local minima. However, often in practice, computing the loss and gradient for the entire training set can be very slow and sometimes intractable on a single machine if the dataset is too big to fit in the main memory.

Stochastic Gradient Descent (SGD) can overcome the cost of GD and still lead to fast convergence. It addresses the issues by following the negative gradient of the objective after seeing only a single training examples.

$$\theta_{k+1} \leftarrow \theta_k - \eta \nabla_\theta L(f(\theta_k, \mathbf{x}_i), y_i)$$

with a pair $(\mathbf{x}_i, y_i)$ from the training set. Generally, each parameter update in SGD is calculated w.r.t a few training examples or a minibatch as opposed to a single example. The reason for this is twofold: first, this reduces the variance in the parameter update and can lead to a more stable convergence, second this allows the computations that should be used in a well-vectorized computation of the cost and gradient. One important point regarding SGD is the order in which we present the data to the algorithm. If the data is given in some meaningful order, this can bias the gradient and lead to poor convergence. A good method to avoid this is to randomly shuffle the data before each epoch of training.

2.5 Convergence of Gradient descent

Since our work is analyzing convergence properties of training DEQ models, in this section, we will provide some convergence analysis of GD under suitable conditions. Consider the basic unconstrained optimization problem 2.10. Denote the optimal value $\min_\theta \mathcal{L}(\theta)$ by $\mathcal{L}^*$. We are interested in whether the GD algorithm can find a global minimum which is a point θ^* such that $\mathcal{L}(\theta^*) = \mathcal{L}^*$. Throughout this section, we assume the map $\mathcal{L}$ is Lipschitz continuous, smooth and the set of minimizer $\boldsymbol{\theta}^*(\mathcal{L}) := \{\theta | \mathcal{L}(\theta) = \mathcal{L}^*\}$ is not empty. For simplicity, we will only consider fixed step-size GD updates.

Lipschitz continuity A direct consequence of the L-Lipschitz condition is that $\|Df(w)\|_2 \leq L$ for all $w \in \mathbb{R}^m$. This can be seen by re-arrange the equation gives $\frac{\|f(u)-f(v)\|}{\|u-v\|} \leq L$, which is approximately the "magnitude" of the Jacobian when u, v are close. In other words, a function is Lipschitz means it does not have sharp changes everywhere.

Definition 2.5.1 (L-Lipschitz). *A map $f : \mathbb{R}^m \mapsto \mathbb{R}^n$ is L-Lipschitz continuous, if*

$$\|f(u) - f(v)\| \leq L\|u - v\|, \forall u, v \in \mathbb{R}^m$$

Smoothness The gradient of a function measures how the function changes when we move in a particular direction from a point. (S)GD can decrease the function's value by moving the direction opposite of first-order gradient of the objective function. But how do we know that this gradient information is informative? If the gradient were to change arbitrarily quickly, the old gradient does not give us much information at all even if we take a small step. The next assumption, smoothness, assures us that the gradient cannot change too quickly. Therefor we have an assurance that the gradient information is informative within a region around where it is taken.

Definition 2.5.2 (L-smooth function)**.** *A function f is L-smooth (L-Lipschitz gradient) if f is differentiable and (Following definitions of L-smooth function are equivalent)*

- *∇f is L-Lipschitz*
- *f is bounded by quadratic function with $L > 0$.*
$$|f(\mathbf{y}) - f(\mathbf{x}) - \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle| \leq \frac{L}{2}\|\mathbf{y} - \mathbf{x}\|_2^2.$$
- *∇f is monotonic with additional term with $L > 0$*
$$\langle x - y, \nabla f(x) - \nabla f(y) \rangle \leq \frac{1}{L}\|\nabla f(x) - \nabla f(y)\|^2$$
- *If f is twice differentiable, $\nabla^2 f(x) \preceq L\mathbf{I}$, or all eigenvalue of $\nabla^2 f(x)$ is upper bounded of L.*

From definition 2.5.2 of the L-smooth function, there are two things to remember about smoothness,

1. A function f is L-smooth if for any two points $x, y \in \mathrm{dom} f$, $f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|y - x\|^2$. For interpretation, f is globally bounded above by a quadratic function. In other words, it provides an upper bound for the change of the function by means of a quadratic (f cannot be "too sharp ").
2. If x^* is an optimal solution, then $\nabla f = 0$, so that smoothness provides an upper bound on the suboptimality of the function value and the distance to optimality: $f(x) - f(x^*) \leq \frac{L}{2}\|x - x^*\|$.

We can now analyse the convergence of GD on L-smooth functions. Let us consider the smoothness inequality at two iterates θ_t and θ_{t+1} in the scheme from gradient descent.

$$\begin{aligned}\mathcal{L}(\theta_t) - \mathcal{L}(\theta_{t+1}) &\geq \eta \langle \nabla \mathcal{L}(\theta_t), \nabla \mathcal{L}(\theta_t) \rangle - \eta^2 \frac{L}{2}\|\nabla \mathcal{L}(\theta_t)\|^2 \\ &= (\eta - \eta^2 \frac{L}{2})\|\nabla \mathcal{L}(\theta_t)\|^2\end{aligned}$$

Since the right-hand side is concave in η. We can maximize it w.r.t η and obtain the optimal choice $\eta^* = \frac{1}{L}$. This lead to the most important thing is that smoothness induces progress in schemes such as decent step. We call this descent property.

Proposition 2. *Gradient Descent method satisfies "descent" property for a step-size* $\eta \leq \frac{1}{L}$ *when g is a L-smooth fucntion*

$$\forall t \geq 0, \forall \eta \leq \frac{1}{L} : \underbrace{\mathcal{L}(\theta_t) - \mathcal{L}(\theta_{t+1})}_{primal\ progress} \geq \frac{\eta}{2}\|\nabla\mathcal{L}(\theta_t)\|_2^2$$

We can use this descent property to analysis convergence rate of gradient descent to ϵ-critical point.

Theorem 2.5.1. *Gradient descent on L-smooth functions, with a fixed step-size of* $\frac{1}{L}$ *achieve an* ϵ *critical point in* $\frac{2L(\mathcal{L}(\theta_0)-\theta^*)}{\epsilon^2}$ *iterations.*

Theorem 2.5.1 guarantees we were able to find an ϵ-critical point using gradient descent while only assuming L-smoothness. This ends up not being very informative because gradients can equal zero at a saddle point or a maximum—these points are certainly not optimal. Let look at second-order Taylor expansion of 2.9, assume $\mathcal{L}(\theta)$ is second differentiable

$$\mathcal{L}(\theta) = \mathcal{L}(\theta_0) + \nabla\mathcal{L}(\theta_0)^\top(\theta - \theta_0) + \frac{1}{2}(\theta - \theta_0)^\top\nabla^2\mathcal{L}(\theta_0)(\theta - \theta_0) \tag{2.12}$$

When the gradients equal zero, i.e at a critical point, the type of this point depends on higher-order gradients which in this case is the Hessian. We temporarily ignore the case of zero eigenvalues (which is also interesting). If $\nabla^2\mathcal{L}(\theta_0)$ has some negative eigenvalue and some positive then we are at a saddle point and we can further decrease the loss by following their eigenvector. Some studies suggest that SGD can escape saddle points. If $\nabla^2\mathcal{L}(\theta_0)$ is negative definite then we have local maxima which are certainly not the case. Often, we have local minima where $\nabla^2\mathcal{L}(\theta_0)$ is positive definite. So our objective is to find settings that guarantee local minima have good properties such as all global minima are global or all strict local minima are global. And further, we require the convergence rate is fast enough. For the guarantees, because gradient descent can only access local information we might need a local property that ensures global optimality. Since it is well-known that convexity implies local minima are global, we first look at the convex, in particular, strongly convexity case to get a sense of how the linear convergence rate can be achieved.

Convergence under strong convexity Convexity will allow us to translate gradient information into the distance from the optimum and suboptimality of the function value. It is represented in the following lemma which is a consequence of a strongly convex function.

Lemma 2.5.2. *For a* μ*-strongly convex function* f*:*

$$\forall w \ \frac{1}{2\mu}\|\nabla f(w)\|_2^2 \geq (f(w) - f^*) \geq \frac{\mu}{2}\|w - w^*\|$$

where $f^* = \min_w f(w)$*.*

So with strong convexity, we can upper bound the suboptimality of the function value, i.e $(\mathcal{L}(\theta) - \mathcal{L}^*)$, which is a global property, by the norm of gradient, which is a local property. We can easily see that lemma 2.5.2 implies local minima are global. In particular, recall that we have defined $\boldsymbol{\theta}^*(\mathcal{L})$ to be the set of minimizers of $\mathcal{L}$, then $\theta^* \in \boldsymbol{\theta}^*(\mathcal{L})$ if and only if $\nabla\mathcal{L}(\theta^*) = 0$. Also, this lemma directly leads to the following theorem.

Theorem 2.5.3. *For a L-smooth and μ-strongly convex function $\mathcal{L}$, a non empty solution set $\boldsymbol{\theta}^*$, then gradient descent minimization process with a step-size of* $1/L$,

$$\theta_{k+1} = x_k - \frac{1}{L}\nabla f(\theta_k),$$

can be upper bound

$$\mathcal{L}(\theta_t) - \mathcal{L}^* \leq (1 - \frac{\mu}{L})^t(\mathcal{L}(\theta_0) - \mathcal{L}^*)$$

With some modification, we have the convergence rate

$$t \geq \frac{L}{\mu}\log(\frac{\mathcal{L}(\theta_0) - \mathcal{L}^*}{\epsilon})$$

This rate is called linear convergence since because it looks linear on a semi-log plot. Note that lemma 2.5.2 also upper bound the distance to the solution which implies that the solution is unique. As we will see later, this condition can be weaken by forgone the uniqueness of the optimum.

Beyond convexity It is easier for GD/other methods to optimize a convex objective function. In contrast, we often think non-convex is bad and it might have many local minima and no general global optimization techniques. It is true that a non-convex problem in general has many spurious local minima, but there are objective functions that are nice even though they are non-convex. The key property of convexity is that they guarantee critical points are good. Next, we will introduce a conditions that basically say the same thing: when the gradient vanishes some good thing happen.

Definition 2.5.3 (Polyak-Lojasiewicz (PL) condition)**.** *A non-negative function $\mathcal{L}$ satisfies μ-PL for $\mu > 0$, if*

$$\|\nabla\mathcal{L}(\theta)\|^2 \geq \mu\mathcal{L}(\theta) \quad \forall\theta \in \mathbb{R}^p$$

One way to interpreted this condition is it requires that the gradient grows faster than a quadratic function as we move away from the optimal function value. Note that this inequality is similar to lemma 2.5.2 in strongly convex, so it also implies every stationary point is a global minimum. But unlike SC, it does not imply that there is a unique solution so it is a weaker condition. It is also straight forward to prove linear convergence for GD using PL condition. We can simply replace lemma 2.5.2 by inequality on 2.5.3 which leads to following theorem.

Theorem 2.5.4. *Consider the problem 2.10 where f is a L-smooth function, satisfies PL condition and a non empty solution set $\boldsymbol{\theta}^*$, then gradient descent minimization process with a step-size of $1/L$,*

$$\theta_{k+1} = \theta_k - \frac{1}{L}\nabla\mathcal{L}\left(\theta_k\right),$$

can be upper bound

$$f(\theta_t) - f^* \leq (1 - \frac{\mu}{L})^t (f(\theta_0) - f^*)$$

2.6 The training dynamic of Neural network

Let us start this section with an analysis of the training dynamic of linear regression. Then later, we will extend this analysis to the dynamic of training neural networks. Note that instead of the gradient dynamics of parameters, we keep track of the change of predictions during training.

The training dynamic of linear regression Consider a linear regression problem with a finite training set $\{\mathbf{x}_i, y_i\}_{i=1}^N$, and the model is assumed to be linear in some feature representation $\phi(\mathbf{x}) \in \mathbb{R}^n$:

$$\hat{y} = \mathbf{w}^\top \phi(\mathbf{x})$$

where $\mathbf{w} \in \mathbb{R}^n$ and we assume bias absorbed into ϕ. As is common practice for linear regression, we initialize $\mathbf{w} = \mathbf{0}$. We train the model by minimizing the mean squared loss on the training set:

$$\begin{aligned} \mathbf{w}^* &\in \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \\ &= \arg\min_{\mathbf{w}} \frac{1}{2N} \|\Phi\mathbf{w} - \mathbf{y}\|^2 \\ &= \arg\min_{\mathbf{w}} \frac{1}{2N} \|\hat{\mathbf{y}} - \mathbf{y}\|^2 \end{aligned}$$

where $\Phi \in \mathbb{R}^{N\times n}$ is a matrix whose ith row is the feature vector $\phi(\mathbf{x}_i)$. Note that if the solution of arg min is not unique which is the case when $n > N$ or in over-paramterize regime, then which optimum we wind up in depends on the dynamics of training. Since we initialize $\mathbf{w} = \mathbf{0}$ and using GD with step size η, i.e

$$\begin{aligned} \mathbf{w}_{k+1} =& \mathbf{w}_k - \eta\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}_k) \\ =& \mathbf{w}_k - \eta\Phi^\top(\hat{\mathbf{y}}(t) - \mathbf{y}), \end{aligned}$$

the evolution of $\hat{\mathbf{y}}$ is following

$$\begin{aligned}\hat{\mathbf{y}}(0) =& 0,\\ \hat{\mathbf{y}}(k+1) =& \hat{\mathbf{y}}(k) - \eta \underbrace{\Phi\Phi^\top}_{K}(\hat{\mathbf{y}}(k) - \mathbf{y}).\end{aligned}$$

note that the matrix $K \in \mathbb{R}^{N\times N}$ is fixed during the training so we have a *linear* update rule and so

$$\hat{\mathbf{y}}(k+1) - \mathbf{y} = (I - \eta K)^{k+1}(\hat{\mathbf{y}}(0) - \mathbf{y}) = -(I - \eta K)^{k+1}\mathbf{y}. \tag{2.13}$$

Theorem 2.6.1 (Spectral theorem)**.** *Any symmetric matrix S has a full set of eigenvectors, the corresponding eigenvalues are all real, and the eigenvectors can be taken to be orthogonal*

To understand the dynamics, we are interested in the eigenvectors and eigenvalues of K. Using the spectral theorem from linear algebra and the fact that $K = \Phi\Phi^\top$ is symmetric, we can expressed it in terms of the spectral decomposition:

$$K = QDQ^\top$$

where Q is an orthogonal matrix whose columns contain the eigenvectors of K and D is a diagonal matrix whose diagonal entries are the corresponding eigenvalues. And since K is positive semi-definite so all the eigenvalues are non-negative. With this decomposition and from 2.13, we have $\hat{\mathbf{y}}(k) - \mathbf{y} = -(I - QDQ^\top)^k\mathbf{y}$. Using the fact that $\mathbf{y} = QQ^\top\mathbf{y}$, we have

$$\begin{aligned}\hat{\mathbf{y}}(k) - \mathbf{y} =& -Q(I - \eta D)^k Q^\top \mathbf{y}\\ =& -\sum_{i=1}^{N}(1 - \eta\lambda_i)^k(\mathbf{v}_i^\top\mathbf{y})\mathbf{v}_i,\end{aligned}$$

where $\mathbf{v}_1, \dots, \mathbf{v}_N$ are orthonormal eigenvectors of K and $\lambda_1, \dots, \lambda_N$ are corresponding eigenvalues.

Here we can

$$\begin{aligned}\|\hat{\mathbf{y}}(k) - \mathbf{y}\|_2 =& \left\| -\sum_{i=1}^{N}(1 - \eta\lambda_i)^k(\mathbf{v}_i^\top\mathbf{y})\mathbf{v}_i \right\|_2\\ =& \sqrt{\sum_{i=1}^{N}(1 - \eta\lambda_i)^{2k}(\mathbf{v}_i^\top\mathbf{y})^2},\end{aligned}$$

This equation helps us analyze the convergence rate of linear regression. For example, if the label set $\mathbf{y}$ aligns with the least-significant eigenvector, the convergence rate of GD will be very slow. This can be shown by decomposing the label vector $\mathbf{y}$ into its projection onto all eigenvectors $\mathbf{v}_i$. And as k increase, each summand in $\sum_{i=1}^{N}(1 - \eta\lambda_i)^{2k}(\mathbf{v}_i^\top\mathbf{y})^2$ decrease at ratio $(1 - \eta\lambda_i)^2$. And if η is small enough such that $\max_{i\in[N]} \eta\lambda_i < 1$, then the larger λ_i is, the faster $(1 - \eta\lambda_i)^{2k}(\mathbf{v}_i^\top\mathbf{y})^2$ decrease to zeros.

The training dynamic of neural networks We can apply the above analysis to neural networks which is a parameterized function. Let call it $f(\theta, \mathbf{x})$ where $\theta \in \mathbb{R}^p$ is all the (trainable) parameters in the network and $\mathbf{x} \in \mathbb{R}^d$ is the input. Given a training dataset $\{\mathbf{x}_i, y_i\}_{i=1}^N \subset \mathbb{R}^d \times \mathbb{R}$ and consider training the neural network by minimizing the squared loss over training data: $\mathcal{L} = \frac{1}{2N}(f(\theta, \mathbf{x}_i) - y_i)^2$. We want to analyze the output during the training i.e for the set $\{\theta_k\}_{k=1}^T$ is the set of parameters reached by GD so

$$\begin{aligned} \theta_0 &\overset{\text{i.i.d}}{\sim} \mathcal{N}(0,1) \\ \theta_{k+1} =& \theta_k - \eta \nabla_\theta \mathcal{L}(\theta_k); \end{aligned}$$

To do the analysis, first, we can naively expand f in θ around a base point θ_0 by using first-order Taylor expansion

$$f(\theta, \mathbf{x}) - f(\theta_0, \mathbf{x}) \approx \langle \nabla_\theta f(\theta, \mathbf{x}), \theta - \theta_0 \rangle$$

which result in

$$\begin{aligned} f(\theta_k, \mathbf{x}) - f(\theta_{k-1}, \mathbf{x}) \approx& \langle \nabla_\theta f(\theta_{k-1}, \mathbf{x}), \theta_k - \theta_{k-1} \rangle \\ =& \langle \nabla_\theta f(\theta_{k-1}, \mathbf{x}), -\eta \nabla_\theta \mathcal{L}(\theta_{k-1}) \rangle \\ =& -\eta \left\langle \nabla_\theta f(\theta_{k-1}, \mathbf{x}), \nabla_{\hat{\mathbf{y}}} \mathcal{L}(\hat{\mathbf{y}}(k-1)) J(k-1)) \right\rangle \\ =& -\frac{\eta}{N} \left\langle \nabla_\theta f(\theta_{k-1}, \mathbf{x}), (\hat{\mathbf{y}}(k-1) - \mathbf{y})^\top J(k-1)) \right\rangle \\ =& -\frac{\eta}{N} \nabla_\theta f(\theta_{k-1}, \mathbf{x}) J(k-1)^\top (\hat{\mathbf{y}}(k-1) - \mathbf{y}) \end{aligned}$$

where $\hat{\mathbf{y}}(k)$ denote all of the outputs $f(\theta_k, \cdot)$ on the entire dataset at GD timestep k, stacked into a vector, and $J \in \mathbb{R}^{N \times p}$ be the Jacobian of $\hat{\mathbf{y}}$ with respect to θ.

For the dynamic of $\hat{\mathbf{y}}$

$$\hat{\mathbf{y}}(k+1) - \hat{\mathbf{y}}(k) \approx -\frac{\eta}{N} J J^\top (\hat{\mathbf{y}}(k) - \mathbf{y})$$

The matrix $K = JJ^\top$ is the *neural tangent kernel (NTK)*. Note that, unlike for linear regression, this dynamic model is only approximate because *(1) we are using first-order Taylor expansion* and *(2) the kernel K is not fixed but changes over time.* So under which assumption/setting can we overcome these obstacles?

For (1), recall that the error of Taylor expansion become smaller as $\|\theta_k - \theta_{k-1}\|_2$ is small. This interpretation becomes exact when we consider the gradient flow, the continuous time limit of gradient descent (i.e. lots of steps with infinitesimally small learning rate) In this regime, the training dynamic of parameters θ is describe by a ODE equation $\frac{d\theta}{dt} = \nabla \mathcal{L}(\theta(t))$. Let $\hat{\mathbf{y}}(t) = [f(\theta(t), \mathbf{x}_i)]_{i=1}^n \in \mathbb{R}^N$ be the network outputs in the continuous time training. The following lemma tells us about the evolution of the predictions on training data points.

Lemma 2.6.2. *Consider minimizing the squared loss $\mathcal{L}(\theta)$ by gradient descent with infinisimally small learning rate: $\frac{d\theta(t)}{dt} = -\nabla_\theta \mathcal{L}(\theta(t))$. Let $\hat{\mathbf{y}} = (f(\theta(t), \mathbf{x}_i)_{i\in[N]} \in \mathbb{R}^N$ be the network outputs on all $\mathbf{x}_i$ at time t, and $\mathbf{y} = (y_i)_{i\in[N]}$ be the desired outputs. Then, during training, $\hat{\mathbf{y}}(t)$ follows the following evolution*

$$\frac{d\hat{\mathbf{y}}(t)}{dt} = -K(t)(\hat{\mathbf{y}}(t) - \mathbf{y}) \tag{2.14}$$

where $K(t)$ is a $n \times n$ PSD matrix whose the entry is $K_{i,j} = \left\langle \frac{\partial f(\theta(t), x_i)}{\partial \theta}, \frac{\partial f(\theta(t), x_j)}{\partial \theta} \right\rangle$

Proof. Since we use GD flow:

$$\begin{aligned}
\frac{d\hat{y}_i(t)}{dt} = \frac{df(\theta(t), \mathbf{x}_i)}{dt} &= \left(\frac{\partial f(\theta(t), \mathbf{x}_i)}{\partial \theta}\right)^T \frac{d\theta(t)}{dt} \\
&= -\sum_{j=1}^N \frac{d\mathcal{L}}{d\hat{y}_i}(\hat{\mathbf{y}}(t)) \left(\frac{\partial f(\theta(t), \mathbf{x}_i)}{\partial \theta}\right)^T \frac{\partial f(\theta(t), \mathbf{x}_j)}{\partial \theta} \\
&= -\sum_{j=1}^N \frac{d\mathcal{L}}{d\hat{y}_i}(\hat{\mathbf{y}}(t)) \left\langle \frac{\partial f(\theta(t), \mathbf{x}_i)}{\partial \theta}, \frac{\partial f(\theta(t), \mathbf{x}_j)}{\partial \theta} \right\rangle \\
&= -\sum_{j=1}^N \frac{d\mathcal{L}}{d\hat{y}_i}(\hat{\mathbf{y}}(t)) K_{ij} \\
&= -K_{i,:}(t) \frac{d\mathcal{L}}{d\hat{\mathbf{y}}}(\hat{\mathbf{y}}(t))
\end{aligned}$$

We have:

$$\frac{d\hat{\mathbf{y}}(t)}{dt} = -K(t) \cdot \frac{d\mathcal{L}}{d\hat{\mathbf{y}}}(\hat{\mathbf{y}}(t)).$$

□

Remark. We can decompose the above proof into 2 step.

1. $\frac{d\theta(t)}{dt}$: Analyze the effect of outputs $\mathbf{u}_i(t)$ to the update of parameters θ.
2. $\frac{d\mathbf{u}_i(t)}{dt} = \left\langle \frac{\partial \mathbf{u}_i(t)}{\partial \theta}, \frac{d\theta(t)}{dt} \right\rangle$: Analyze the effect of updating θ to the output of the network at $\mathbf{x}_i$.

By combining these 2 step, we have the relation of the output at $\mathbf{x}_i$ and the output at other examples in training set S. These analysis can extend to the any inputs $\mathbf{x}$ outside of training dataset.

$$\begin{aligned}\frac{df(\theta(t),\mathbf{x})}{dt} &= \left\langle \frac{\partial f(\theta(t),\mathbf{x})}{\partial\theta}, \frac{d\theta(t)}{dt} \right\rangle \\ &= -\frac{1}{N}\sum_{j=1}^{N} \frac{dL}{du_j}(\mathbf{u}(t)) \left\langle \frac{\partial f(\theta(t),\mathbf{x})}{\partial\theta}, \frac{\partial f(\theta(t),\mathbf{x}_j)}{\partial\theta} \right\rangle\end{aligned}$$

We define the tangent kernel $K_{\mathbf{x},\mathbf{x}'}(\theta)$ for fixed inputs $\mathbf{x},\mathbf{x}' \in \mathbb{R}^d$ as follows

$$K_{\mathbf{x},\mathbf{x}'}(\theta) \coloneqq \nabla_\theta f(\theta;\mathbf{x})^\top \nabla_\theta f(\theta;\mathbf{x}')$$

It is important to note that the tangent kernel is associated to the network itself and does not depend on the optimization algorithm or the choice of a loss function, which are parts of the training process. There are many studies have shown that as the width of certain neural networks increases, the kernel $K_{\mathbf{x},\mathbf{x}'}(\theta)$ is a constant function of the weight during training, result in an approximately linear training dynamic. This enables us to use NTK to prove the global convergence of GD for neural networks.

2.6.1 Prove convergence of neural network using NTK

In this section, we will consider conditions on NTK which guarantee the global convergence of GD. Let define $F(\theta) \triangleq (f(\theta,\mathbf{x}_1),\ldots,f(\theta,\mathbf{x}_N))^\top$ and the matrix-output function $K(\theta) : \mathbb{R}^p \mapsto \mathbb{R}^{N\times N}$ where $K(\theta)_{ij} = K_{\mathbf{x}_i,\mathbf{x}_j}(\theta)$. For simplicity, we will assume $F(\theta)$ is L-Lipschitz continuous and β-smooth.

As discuss in section 2.5, PL condition leads to global convergence of GD. However, we might not need to have the PL condition for all the domain of $\mathcal{L}$ but only in a region that is reachable by the GD algorithm. This leads to a variant of the PL condition which is also introduced in Kawaguchi (2021); Liu et al. (2020).

Definition 2.6.1 (PL* condition). *A non-negative function $\mathcal{L}$ satisfies μ-PL* on a set $\mathcal{S} \subset \mathbb{R}^p$ for $\mu > 0$, if*

$$\|\nabla\mathcal{L}(\theta)\|^2 \geq \mu\mathcal{L}(\theta) \quad \forall\theta\in\mathcal{S}$$

where we assume there exist a solution of $\mathcal{L}(\theta) = 0$.

Now, we state following theorem from Liu et al. (2020) which show that PL* condition in a ball is sufficient for global convergence.

Theorem 2.6.3 (Informal version of Theorem 6 in Liu et al. (2020)). *Suppose $F(\theta)$ is L-Lipschitz continuous and β-smooth. If the square loss $\mathcal{L}(\theta)$ satisfies the μ-PL* condition in the ball $B(\theta(0),R) \triangleq \{\theta\in\mathbb{R}^p : \|\theta-\theta(0)\| \leq R\}$ with $\mathbb{R} = c\sqrt{\mathcal{L}(\theta(0))}$. Then GD with a step size small enough step size η convergence to a global minima in $B(\theta(0),R)$ with an linear convergence rate*

$$\mathcal{L}(\theta(k)) \leq (1-\kappa)^k\mathcal{L}(0)$$

where c,κ is positive constant and $\kappa < 1$.

Now we consider a condition on NTK called uniform conditioning. This condition is introduced in Liu et al. (2020).

Definition 2.6.2 (Uniform conditioning (Liu et al., 2020)). *We say that $F(\theta)$ is μ-uniformly conditioned $\mu > 0$ on $S \subset \mathbb{R}^p$ if the smallest eigenvalue of its tangent kernel $K(\theta)$ satisfies*

$$\lambda_{\min}(K(\theta)) \geq \mu, \forall w \in S.$$

The following theorem connect uniform condition and PL* condition.

Theorem 2.6.4 (Uniform conditioning $\Rightarrow$ PL* condition. (Theorem 1 in Liu et al. (2020))). *If $F(\theta)$ is μ-uniformly conditioned, on a set $S \subset \mathbb{R}^p$, then the square loss $\mathcal{L}(\theta) = \frac{1}{2}\|F(\theta) - \mathbf{y}\|^2$ satisfies μ-PL* condition on $\mathcal{S}$.*

So from theorem 2.6.3 and 2.6.4, we can guarantee global convergence of GD by lower bound $\lambda_{\min}(K(\theta))$ in a ball $B(\theta(0), R)$. This can be done by lower bound the minimum eigentvalue of the kernel at its initialization i.e $\lambda_{\min}(K(\theta(0))$ then invokes Weyl's inequality which state that

$$\lambda_{\min}(K(\theta)) \geq \lambda_{\min}(K(\theta(0)) - \|K(\theta) - K(\theta(0)\|.$$

For the perturbation term, we relies on following analysis on trajectory length of the parameter θ.

PL* condition $\Rightarrow$ Bound trajectory length Let $\mathcal{L}(\theta)$ is a β-smooth objective function and satisfies μ-PL* condition on $B(\theta_0, R)$ and where $R = c\sqrt{\mathcal{L}(\theta_0)}$. We will prove by induction that the trajectory of GD stay in the ball $B(\theta_0, R)$. Assume $\theta \in \mathcal{S}$ for timestep $s \in [k]$, and step size $\eta \leq \frac{1}{\beta}$. We have

$$\|\theta(k+1) - \theta(0)\| \leq \sum_{i=0}^{k} \|\theta(i+1) - \theta(i)\|$$

$$\leq 2\sqrt{\frac{\mathcal{L}(\theta(0))}{\mu}} = R$$

where the proof for the last inequality we refer to Gupta et al. (2019). So $\theta(k+1)$ resides in the ball $B(\theta(0), R)$. This conclude our induction and prove that if $\mathcal{L}$ satisfied μ-PL* condition in a ball $B(\theta(0), R)$ then the optimization path $\{\theta(i)\}_{i=0}^{\infty}$ stay in $B(\theta(0), R)$.

From this result, we can do some analysis and upper bound the term $\|K(\theta) - K(\theta(0)\|$ for $\theta \in B(\theta(0), R)$.

Chapter 3

On the optimization theory of Monotone DEQ

In this chapter, we will use NTK tools to analyze optimization of DEQ models. First, we present our observation about loss landscape of DEQ models by using visualization technique. From this observation, we state our conjecture about optimizing DEQ models. Next, we will apply NTK to analyze the training of DEQ models and prove our conjecture for monDEQ models which is a well-pose guaranteed class of DEQ models (as discussed in 2.3).

3.1 An observation of loss landscape of DEQ models

We start off by "taking a look" on the underlying DEQ's loss landscapes by visualizing the loss function around the last checkpoint. The detail of this experiment is present in section 4.2. As in Fig 3.1 which is the visualization of loss landscape of MDEQ model, we observe that the loss landscape has only one minima and this minima is flat. This observation suggest that MDEQ models might satisfied PL condition around the optimum and attain a global convergence this region (as discuss in section 2.5). From this observation, we state the following conjecture about the optimization of DEQ models.

Conjecture 1. *Under suitable conditions, optimizing DEQ model using (S)GD algorithm attain linear convergence rate*

In the following sections, we will find conditions under which GD achieve linear convergence rate when train a variant of DEQ models called Monotone DEQ which have been introduce in chapter 2. Monotone DEQs (monDEQs) parameterize DEQ in the manner that guarantee the existence of a unique fixed point.

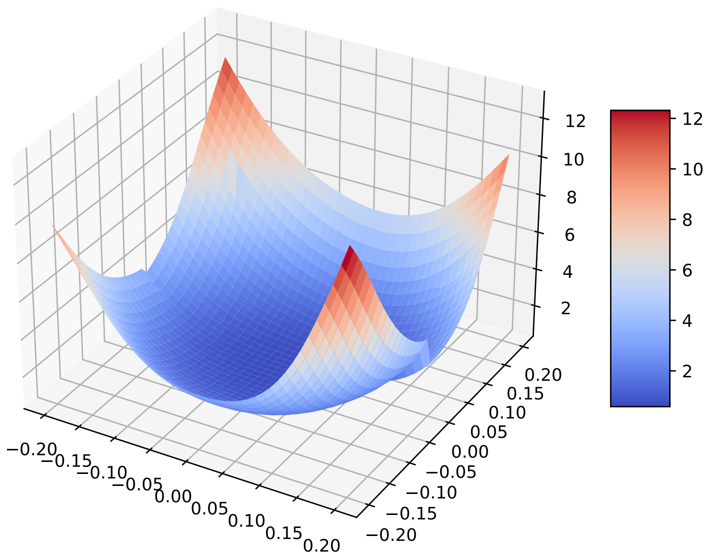


Figure 3.1: Loss landscape of trained MDEQ model on 1024 exsamples from ImageNet test dataset.

3.2 Priliminaries

3.2.1 Setting: Monotone DEQ Trained by Gradient Descent

We consider a fully connected monotone DEQ ReLU activated network with n hidden neurons in the hidden layers:

Definition 3.2.1 (Monotone DEQ). *The monDEQ consider a weight-tied, input injected network with iterations of the form:*

$$z^{[k+1]} = \sigma\left(W\mathbf{z}^{[k]} + U\mathbf{x}\right) \tag{3.1}$$

where $\mathbf{x} \in \mathbb{R}^d$ is the input, $U \in \mathbb{R}^{n\times d}$ is the input-injection weights, $W \in \mathbb{R}^{n\times n}$ is the hidden-unit weights, $\mathbf{z}^{(k)}$ is the hidden unit activation and $\sigma : \mathbb{R} \mapsto \mathbb{R}$ an elementwise non-linearity which is ReLU function in our case. The output of implicit layers is define as the fixed point of the above update function

$$\mathbf{z}^* = \sigma\left(W\mathbf{z}^* + U\mathbf{x}\right) \tag{3.2}$$

To ensure the strong monotonicity condition, that $I - W \succeq mI$, the monDEQ parameterizes W as

$$W = (1-m)I - A^T A + B - B^T. \tag{3.3}$$

The output of the network is define as the linear transformation of the fixed point:

$$f(\theta, \mathbf{x}) = \mathbf{u}^\top \mathbf{z}^*$$

where $\theta = (\mathbf{u}, A, B, U)$ are trainable parameters.

From an underlying data distribution $\mathcal{D}$ over $\mathbb{R}^d \times \mathbb{R}$, we draw i.i.d N input-label samples denoted $\mathcal{S} = \{(\mathbf{x}_i, y_i\}_{i=1}^N$. We also denote $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d\times N}$ and $\mathbf{y} = (y_1, \ldots, y_N)^\top \in \mathbb{R}^N$.

We focus on the empirical risk minimization (ERM) problem over data $\mathcal{S}$ with the quadratic loss function

$$\min_\theta \mathcal{L}(\theta) = \frac{1}{2}\sum_{i=1}^N (f(\theta, \mathbf{x}_i) - y_i)^2 \tag{3.4}$$

where θ is the parameter we optimize over and f is the prediction function, which in our case is a Monotone DEQ model. We train the model by gradient descent (GD) and consider the GD update $\theta(k+1) = \theta(k+1) - \eta\nabla\mathcal{L}(\theta(k+1))$, where η is the learning rate and $\theta(k+1) = \text{vec}[A(k), B(k), U(k), \mathbf{u}(k)]$ is the paraneter we optimize over at step k. Throughout this chapter, we use k as the GD iteration number, and also use k to index all variables that depend on $\theta(k)$. Note that we use $\cdot(k)$ for GD step and $\cdot^{[\ell]}$ for the fixed point solver step which is forward-backward splitting iteration as presented follow.

3.2.2 Forward-backward splitting iteration to estimate lipschitz constant

As discussed in chapter 2, one of the important properties to prove the convergence of GD is the Lipschitz constant of the network. In this preliminary section, we will present part of the work of Pabbaraju et al. (2021) which upper bound this constant for monDEQ models. The idea is instead of naive unrolling forward computation of DEQ model, we unroll steps computed by spliting methods, particularly the forward backward splitting method.

$$\mathbf{z}^{[k+1]} = \sigma\left(\mathbf{z}^{[k]} - \alpha\left((I-W)\mathbf{z}^{[k]} - U\mathbf{x}\right)\right) = \sigma\left(\underbrace{(I - \alpha(I-W))}_{T_\alpha}\mathbf{z}^{[k]} + \alpha U\mathbf{x}\right) \tag{3.5}$$

The operator T_α is contractive for any $\alpha \in (0, \frac{2m}{L^2}]$, and this iteration is guarantee to converge as long as the operator $I - W$ is Lipschitz and strongly monotone with parameters L (which is in fact the spectral norm $\|I - W\|_2$). Denote $L[T_\alpha]$ the lipschitz constant of T_α, we have

Proposition 3. *$L[T_\alpha] \leq \sqrt{1 - 2\alpha m + \alpha^2 L[I-W]^2}$*

This implies that for $\alpha \in \left(0, \frac{2m}{L[I-W]^2}\right)$ then $L[T_\alpha] < 1$.

Additional notation We analyze monDEQ models by unrolling the forward-backward iteration. Here we define the transform at the ℓ-th iteration as

$$Z^{[\ell]} = \sigma(T_\alpha Z^{[\ell-1]} + \alpha U X) \tag{3.6}$$

where $Z^{[\ell]} \in \mathbb{R}^{n\times N}$ is the output feature at the ℓ-th iteration. The output of the monDEQ layer is defined by $Z \triangleq \lim_{\ell\to\infty} Z^{[\ell]}$. We also define $\hat{y}_i = f(\theta, \mathbf{x}_i)$ as the network's prediction on the i-th input and use $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_N)^\top \in \mathbb{R}^N$ to denote all N prediction. Then we have $\hat{\mathbf{y}} = Z^\top \mathbf{u}$ and $\mathcal{L}(\theta) = \frac{1}{2}\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2$. Next we will use these notation to derive the gradient for monotone DEQ models.

3.2.3 Gradient Computation

We apply the theory which have been provide in section 2.1 to derive the gradient for monotone DEQ network. Recall that the equilibrium point of 3.6 is also the solution of $g_\theta(Z;X) \triangleq \; Z - \sigma\left(WZ + UX\right)$. First, we need to derive the gradient of g for each parameters.

Lemma 3.2.1. *The partial derivatives of g_θ with respect to $Z, A, B,$ and U are*

$$\begin{aligned}
\frac{\partial g_\theta}{\partial \operatorname{vec}[Z]} &= \left[I_{Nn} - D\left(I_N \otimes W\right)\right],\\
\frac{\partial g_\theta}{\partial \operatorname{vec}[A]} &= \frac{\partial g_\theta}{\partial \operatorname{vec}[W]}\frac{\partial \operatorname{vec}[W]}{\partial \operatorname{vec}[A]} = -D\left[Z^\top \otimes I_n\right] P_A\\
\frac{\partial g_\theta}{\partial \operatorname{vec}[B]} &= \frac{\partial g_\theta}{\partial \operatorname{vec}[W]}\frac{\partial \operatorname{vec}[W]}{\partial \operatorname{vec}[B]} = -D\left[Z^\top \otimes I_n\right] P_B\\
\frac{\partial g_\theta}{\partial \operatorname{vec}[U]} &= -\,D\left[X^\top \otimes I_n\right]
\end{aligned}$$

.

where $D \triangleq \operatorname{diag}\left[\operatorname{vec}\left(\sigma'(WZ + UX)\right)\right]$, $P_A = -(I_n + K^{(n)})(I_n \otimes A^T)$, $P_B = (I_n - K^{(n)})$ and K is a commutation matrix.

Proof on C.0.1.

Next, we derive the graident of the fixed points Z w.r.t parameters.

Lemma 3.2.2. *The partial derivatives of Z with respect to $A, B,$ and U are*

$$\begin{aligned}
\frac{\partial \operatorname{vec}[Z]}{\partial \operatorname{vec}[W]} &= Q^{-1} D\left[Z^\top \otimes I_n\right],\\
\frac{\partial \operatorname{vec}[Z]}{\partial \operatorname{vec}[A]} &= Q^{-1} D\left[Z^\top \otimes I_n\right] P_A\\
\frac{\partial \operatorname{vec}[Z]}{\partial \operatorname{vec}[B]} &= Q^{-1} D\left[Z^\top \otimes I_n\right] P_B\\
\frac{\partial \operatorname{vec}[Z]}{\partial \operatorname{vec}[U]} &= Q^{-1} D\left[X^\top \otimes I_n\right]
\end{aligned}$$

where $Q \triangleq \frac{\partial g_\theta}{\partial \operatorname{vec}[Z]} = \left[I_{Nn} - D\left(I_N \otimes W\right)\right] \in \mathbb{R}^{Nn\times Nn}$.

Finally, the gradient derivations of the loss function for each trainable parameters are given in lemma 3.2.3

Lemma 3.2.3. *The partial derivatives of $\mathcal{L}$ with respect to Z, A, B, U, and $\mathbf{u}$ are*

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \mathbf{u}} &= (\hat{\mathbf{y}} - \mathbf{y})^\top Z^\top \\
\frac{\partial \mathcal{L}}{\partial A} &= (\hat{\mathbf{y}} - \mathbf{y})^\top \left(\mathbf{u}^\top \otimes I_N\right) Q^{-1} D \left[Z^\top \otimes I_n\right] P_A \\
\frac{\partial \mathcal{L}}{\partial B} &= (\hat{\mathbf{y}} - \mathbf{y})^\top \left(\mathbf{u}^\top \otimes I_N\right) Q^{-1} D \left[Z^\top \otimes I_n\right] P_B \\
\frac{\partial \mathcal{L}}{\partial U} &= (\hat{\mathbf{y}} - \mathbf{y})^\top \left(\mathbf{u}^\top \otimes I_N\right) Q^{-1} D \left[X^\top \otimes I_n\right]
\end{aligned}$$

3.3 Main results

3.3.1 Continuous time analysis

In this section we follow previous works on gradient dynamics of neural networks and consider the process of learning monDEQ models via gradient flow. This is equivalent to the gradient descent with an infinitesimal step-size, i.e

$$\begin{aligned}
\frac{d}{dt} A(t) &= -\frac{\partial \mathcal{L}}{\partial A}(\theta(t)), \quad \forall t \geq 0 \\
\frac{d}{dt} B(t) &= -\frac{\partial \mathcal{L}}{\partial B}(\theta(t)), \\
\frac{d}{dt} U(t) &= -\frac{\partial \mathcal{L}}{\partial U}(\theta(t)), \\
\frac{d}{dt} \mathbf{u}(t) &= -\frac{\partial \mathcal{L}}{\partial \mathbf{u}}(\theta(t)).
\end{aligned}$$

As discussed in 2, our continuous time analysis relies on the analysis of the dynamics of prediction $\hat{\mathbf{y}}(t)$. We derive the dynamics of $\hat{\mathbf{y}}$ in the following lemma.

Proposition 4. *The dynamics of prediction $\hat{\mathbf{y}}$ is given by*

$$\frac{d\hat{\mathbf{y}}}{dt} = -K(t)(\hat{\mathbf{y}} - \mathbf{y}) \tag{3.7}$$

where

$$\begin{aligned}
K(t) &\triangleq J_A(t)^\top J_A(t) + J_B(t)^\top J_B(t) + J_U(t)^\top J_U(t) + Z(t)^\top Z(t) \\
J_A(t) &\triangleq \left(\mathbf{u}(t)^\top \otimes I_N\right) Q(t)^{-1} D(t) \left[Z(t)^\top \otimes I_n\right] P_A(t), \\
J_B(t) &\triangleq \left(\mathbf{u}(t)^\top \otimes I_N\right) Q(t)^{-1} D(t) \left[Z(t)^\top \otimes I_n\right] P_B, \\
J_U(t) &\triangleq \left(\mathbf{u}(t)^\top \otimes I_N\right) Q(t)^{-1} D(t) \left[X^\top \otimes I_n\right].
\end{aligned}$$

Let $\lambda_0 = \sigma_{\min}(Z(0))$. Let define a ball around each parameters called $\mathcal{B}(A(0), R_A)$, $\mathcal{B}(B(0), R_B)$, $\mathcal{B}(U(0), R_U)$ and $\mathcal{B}(\mathbf{u}(0), R_u)$, and let $\bar{\lambda}_\cdot = \| \cdot_0 \|_2 + R_\cdot$ which is the upper bound on the operator norm of each parameters. For convenient, we also denote $\bar{\lambda}_{ab} \triangleq \bar{\lambda}_a \bar{\lambda}_b$ for $a, b \in \{A, B, U, \mathbf{u}\}$. We state the result of convergence of gradient flow in the theorem 3.3.1.

Condition 1. *We define following condition on* λ_0

$$\lambda_0^2 \geq 4\|X\|_F \sqrt{2\mathcal{L}(\theta_0)} \max\left\{ \frac{2\bar{\lambda}_{U\mathbf{u}A}}{m^2 R_A}, \frac{2\bar{\lambda}_{U\mathbf{u}}}{m^2 R_B}, \frac{\bar{\lambda}_{\mathbf{u}}}{m R_U}, \frac{\bar{\lambda}_U}{m R_u} \right\} \tag{3.8}$$

$$\lambda_0^3 \geq 8\|X\|_F^2 \sqrt{2\mathcal{L}(\theta_0)} \frac{\bar{\lambda}_{\mathbf{u}}}{m^2} \left(\frac{4(\bar{\lambda}_{UA}^2 + \bar{\lambda}_U^2)}{m^2} + 1 \right) \tag{3.9}$$

$$\tag{3.10}$$

Theorem 3.3.1. *If λ_0 satisfies condition 1 and the width of hidden layer is $O(N)$ then we have linear convergence*

$$\mathcal{L}(\theta(t)) \leq \exp\{-\lambda_0^2 t/2\} \mathcal{L}(\theta(0))$$

Proof. Let us state some useful inequalities.

Lemma 3.3.2. $\|(\mathbf{u}^\top \otimes I_N) Q^{-1}\| \leq \frac{\|\mathbf{u}\|}{m}$

Proof on C.0.4 Following lemma bounds the norm of feature matrix Z,

Lemma 3.3.3. $\|Z^*\|_F \leq \frac{1}{m}\|X\|_F \|U\|_2$.

Proof on C.0.3.

Let define a ball for each parameters at their initialization called $\mathcal{B}(A(0), R_A)$, $\mathcal{B}(B(0), R_B)$, $\mathcal{B}(U(0), R_U)$ and $\mathcal{B}(\mathbf{u}(0), R_u)$, and let $\bar{\lambda}_\cdot = \| \cdot_0 \|_2 + R_\cdot$ which is the upper bound on the operator norm of each parameters in these ball. We show by induction for all $0 \leq s < t$ that

1. $\cdot_s$ is within the ball $\mathcal{B}(\cdot_0, R_\cdot)$,
2. $\mathcal{L}(\theta(s)) \leq \exp\{-\lambda_0^2 s/2\} \mathcal{L}(\theta(0))$

Suppose that, for a given $k \geq 0$, $\cdot_k$ is in the ball $\mathcal{B}(\cdot_0, R_\cdot)$ and equation 3.20 holds.

$$\begin{aligned} \|\nabla_A \mathcal{L}(\theta(s))\| &\leq \frac{2}{m^2}\|X\|_F \|U(s)\|_2 \|\mathbf{u}\|_2 \|A(s)\|_2 \|\hat{\mathbf{y}}(s) - \mathbf{y}\|_2 \\ &\leq \frac{2}{m^2}\|X\|_F \lambda_{U\mathbf{u}A} \exp\{-(\lambda_0/4)s\} \|\hat{\mathbf{y}}(0) - \mathbf{y}\|_2 \end{aligned}$$

Then

$$\|A(t) - A(0)\| \leq \int_0^t \|\nabla_A \mathcal{L}(\theta(s))\| \leq \frac{8}{\lambda_0^2 m^2}\|X\|_F \lambda_{U\mathbf{u}A} \|\hat{\mathbf{y}}(0) - \mathbf{y}\|_2 \leq R_A$$

Here we apply a condition on λ_0 in the last inequality.

$$\lambda_0^2 \geq \frac{4}{m}\|X\|_F\|\hat{\mathbf{y}}_0 - \mathbf{y}\|_F \max\left\{\frac{2\bar{\lambda}_{U\mathbf{u}A}}{mR_A}, \frac{2\bar{\lambda}_{U\mathbf{u}}}{mR_B}, \frac{\bar{\lambda}_{\mathbf{u}}}{R_U}, \frac{\bar{\lambda}_U}{R_u}\right\}$$

So

$$\|A(t)\| \leq \|A(0)\| + \|A(t) - A(0)\| \leq \bar{\lambda}_A$$

Similarly,

$$\|B(t) - B(0)\|_F \leq \frac{8}{\lambda_0^2}\frac{1}{m^2}\|X\|_F\bar{\lambda}_U\bar{\lambda}_u\|\hat{\mathbf{y}}_0 - \mathbf{y}\|_F \leq R_B \tag{3.11}$$

$$\|U(t) - U(0)\|_F \leq \frac{4}{\lambda_0^2}\frac{1}{m}\|X\|_F\bar{\lambda}_u\|\hat{\mathbf{y}}_0 - \mathbf{y}\|_F \leq R_U \tag{3.12}$$

$$\|\mathbf{u}(t) - \mathbf{u}(0)\|_F \leq \frac{4}{\lambda_0^2}\frac{1}{m}\|X\|_F\bar{\lambda}_u\|\hat{\mathbf{y}}_0 - \mathbf{y}\|_F \leq R_{\mathbf{u}} \tag{3.13}$$

So the induction assumption (1) hold for t and we also have

$$\begin{aligned}\|B(t)\|_2 &\leq \|B(0)\| + R_B = \bar{\lambda}_B\\ \|U(t)\|_2 &\leq \|U(0)\| + R_U = \bar{\lambda}_U\\ \|\mathbf{u}(t)\|_2 &\leq \|\mathbf{u}(0)\| + R_u = \bar{\lambda}_u\end{aligned}$$

Next, we will prove the second induction hypothesis holds for t. Note that

$$\begin{aligned}\frac{d\mathcal{L}(\theta(t))}{dt} =& \frac{d}{dt}\frac{1}{2}\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2\\ =& -(\hat{\mathbf{y}} - \mathbf{y})^\top K(t)(\hat{\mathbf{y}} - \mathbf{y})\\ \leq& -\lambda_{\min}(K(t))\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2\\ =& -2\lambda_{\min}(K(t))\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2\end{aligned}$$

Solving the above ODE yields

$$\mathcal{L}(\theta(t)) \leq \exp\{-2\lambda_{\min}(K(t))t\}\mathcal{L}(\theta(0))$$

Also observe that

$$\lambda_{\min}(K(t)) \geq \lambda_{\min}(Z(t)^\top Z(t)) = \sigma_{\min}(Z(t))^2. \tag{3.14}$$

where we need our assumption $n \geq N$ to optain $\lambda_{\min}(Z(t)^\top Z(t)) = \sigma_{\min}^2(Z(t))$.

So we just need to lower bound the minimum singular value of $Z(t)$. We can do this by upper bound the distance between $Z(t)$ and $Z(0)$ and using Weyl's inequality. First, we derive the dynammic of Z induced by gradient flow,

$$\begin{aligned}
\frac{d\,\mathrm{vec}[Z]}{dt} =& \frac{\partial\,\mathrm{vec}[Z]}{\partial\,\mathrm{vec}[A]}\frac{d\,\mathrm{vec}[A]}{dt} + \frac{\partial\,\mathrm{vec}[Z]}{\partial\,\mathrm{vec}[B]}\frac{d\,\mathrm{vec}[B]}{dt} + \frac{\partial\,\mathrm{vec}[Z]}{\partial\,\mathrm{vec}[U]}\frac{d\,\mathrm{vec}[U]}{dt} \\
=& \frac{\partial\,\mathrm{vec}[Z]}{\partial\,\mathrm{vec}[A]}\left(-\frac{\partial\mathcal{L}}{\partial\,\mathrm{vec}[A]}\right)^\top + \frac{\partial\,\mathrm{vec}[Z]}{\partial\,\mathrm{vec}[B]}\left(-\frac{\partial\mathcal{L}}{\partial\,\mathrm{vec}[B]}\right)^\top + \frac{\partial\,\mathrm{vec}[Z]}{\partial\,\mathrm{vec}[U]}\left(-\frac{\partial\mathcal{L}}{\partial\,\mathrm{vec}[U]}\right)^\top \\
=& Q^{-1}D\left[Z^\top \otimes I_n\right](P_AP_A + P_BP_B)\left[Z\otimes I_n\right]DQ^{-\top}(\mathbf{u}\otimes I_N)(\hat{\mathbf{y}}-\mathbf{y}) \\
&+ Q^{-1}D\left[X^\top\otimes I_n\right]\left[X\otimes I_n\right]DQ^{-\top}(\mathbf{u}\otimes I_N)(\hat{\mathbf{y}}-\mathbf{y})
\end{aligned}$$

Since $P_AP_A = (I+K^{(n)})(A^\top A \oplus A^\top A)$ and $P_BP_B = 2(I-K^{(n)})$ and $\|I+K^{(n)}\|_2 = \|I-K^{(n)}\|_2 = 2$ where $\oplus$ denote Kronecker sum, we have

$$\begin{aligned}
&\left\|\left[Z^\top\otimes I_n\right](P_AP_A+P_BP_B)\left[Z\otimes I_n\right]\right\| \\
=&\left\|\left[Z^\top\otimes I_n\right]((I+K^{(n)})(A^\top A\oplus A^\top A)+2(I-K^{(n)}))\left[Z\otimes I_n\right]\right\| \\
\le&\frac{4\|X\|_F^2(\bar{\lambda}_{UA}^2+\bar{\lambda}_U^2)}{m^2}
\end{aligned}$$

So

$$\begin{aligned}
\left\|\frac{dZ(s)}{dt}\right\| \le& \frac{\bar{\lambda}_{\mathbf{u}}}{m^2}\|X\|_F^2\left(\frac{4(\bar{\lambda}_{UA}^2+\bar{\lambda}_U^2)}{m^2}+1\right)\|\hat{\mathbf{y}}(s)-\mathbf{y}\| \\
\le& \frac{\bar{\lambda}_{\mathbf{u}}}{m^2}\|X\|_F^2\left(\frac{4(\bar{\lambda}_{UA}^2+\bar{\lambda}_U^2)}{m^2}+1\right)\exp\{-(\lambda_0/4)s\}\|\hat{\mathbf{y}}(0)-\mathbf{y}\|
\end{aligned}$$

Now we can upper bound the distant

$$\begin{aligned}
\|Z(t)-Z(0)\| \le \|Z(t)-Z(0)\|_F \le& \int_0^t\left\|\frac{dZ(s)}{dt}\right\| \\
\le& \frac{4\bar{\lambda}_{\mathbf{u}}}{\lambda_0^2 m^2}\left(\frac{4(\bar{\lambda}_{UA}^2+\bar{\lambda}_U^2)}{m^2}+1\right)\|X\|_F^2\|\hat{\mathbf{y}}(0)-\mathbf{y}\| \\
\le& \frac{\lambda_0}{2}
\end{aligned}$$

Weyl's inequality implies that

$$\sigma_{\min}(Z(t)) \ge \sigma_{\min}(Z(0)) - \|Z(t)-Z(0)\| \ge \lambda_0/2$$

Plug this into yields

$$\mathcal{L}(\theta(t)) \le \exp\{-\lambda_0^2 t/2\}\mathcal{L}(\theta(0))$$

□

3.3.2 Discrete time Analysis

Next, we consider discrete gradient descent with step size η. As discussed in chapter 2, we don't have an exact formula of the dynamics of the output $\hat{\mathbf{y}}$ in the discrete time analysis. Instead, we consider the change of $\hat{\mathbf{y}}_k$ between 2 consecutive GD step. First, we observe that the change of the equilibrium point and the output of monDEQ networks is upper bound by the perturbation of their parameter which is show in lemma 3.3.4.

Lemma 3.3.4. *For each $k, s \in [0, T]$, it holds that*

$$\|Z(k) - Z(s)\|_F \leq \|X\|_F \left(\frac{\|U(k) - U(s)\|_2}{m} + \frac{\|W(k) - W(s)\|_2 \|U(s)\|_2}{m^2} \right) \tag{3.15}$$

and

$$\begin{aligned}\|\hat{\mathbf{y}}(k) - \mathbf{y}(s)\| \leq & \|X\|_F \frac{\|\mathbf{u}(k)\| \|U(k) - U(s)\|_2}{m} \\ & + \|X\|_F \frac{\|\mathbf{u}(k)\| \|W(k) - W(s)\|_2 \|U(s)\|_2}{m^2} \\ & + \|X\|_F \frac{\|U(s)\| \|\mathbf{u}(k) - \mathbf{u}(s)\|_2}{m}\end{aligned}$$

The proof is deferred in Appendix C.0.5. This lemma enable us to use a small enough step size to diminishing the perturbation term in the loss function. Using same notation as in continuous analysis part (3.3.1), we provide the convergence result of GD in the following theorem:

Condition 2. *We define following condition on λ_0*

$$\lambda_0^2 \geq 8\|X\|_F \sqrt{2\mathcal{L}(\theta_0)} \max \left\{ \frac{2\bar{\lambda}_{U\mathbf{u}A}}{m^2 R_A}, \frac{2\bar{\lambda}_{U\mathbf{u}}}{m^2 R_B}, \frac{\bar{\lambda}_{\mathbf{u}}}{m R_U}, \frac{\bar{\lambda}_U}{m R_u} \right\} \tag{3.16}$$

$$\lambda_0^3 \geq 8\|X\|_F^2 \sqrt{2\mathcal{L}(\theta_0)} \bar{\lambda}_{\mathbf{u}} \left(1 + 6\bar{\lambda}_{UA}^2 + 4\bar{\lambda_U}^2 \right) \tag{3.17}$$

$$\lambda_0^2 \geq 20\|X\|_F^2 \left(\bar{\lambda}_u^2 + 4\bar{\lambda}_{uUA}^2 + 4\bar{\lambda}_{uU}^2 + \bar{\lambda}_U^2 \right) \tag{3.18}$$

Theorem 3.3.5. *If λ_0 satisfies condition 2 and the width of hidden layer is $O(N)$ and small enough step size η s.t*

$$\eta \leq \min\{\frac{4}{\lambda_0^2}, \|X\|_F^{-2}(\bar{\lambda}_u^2 + 4\bar{\lambda}_{Uu}^2 + 4\bar{\lambda}_{UuA}^2 + \bar{\lambda}_U^2)^{-1}, \frac{1}{2}\|X\|_F^{-1}\lambda_0^{-1}\bar{\lambda}_{UuA}^{-1}, \frac{1}{6}\lambda_0^{-2}\bar{\lambda}_{uA}^{-2}\}$$

then we have linear convergence

$$\mathcal{L}(k+1) \leq \mathcal{L}(\theta_k) \left[1 - \eta \frac{\lambda_0^2}{4} \right]$$

Proof sketch We write the loss as

$$\mathcal{L}(\theta(k)) = \frac{1}{2}\|\hat{\mathbf{y}}(k) - \mathbf{y}\| \tag{3.19}$$

We consider one iteration on the loss function

$$\begin{aligned}
&\mathcal{L}(\theta(k+1)) \\
=&\frac{1}{2}\|\hat{\mathbf{y}}(k+1) - \mathbf{y}\|_2^2 \\
=&\frac{1}{2}\|\hat{\mathbf{y}}(k+1) - \hat{\mathbf{y}}(k) + \hat{\mathbf{y}}(k) - \mathbf{y}\|_2^2 \\
=&\frac{1}{2}\|\hat{\mathbf{y}}(k+1) - \hat{\mathbf{y}}(k)\|_2^2 + \left(\hat{\mathbf{y}}(k+1) - \hat{\mathbf{y}}(k)\right)^\top \left(\hat{\mathbf{y}}(k) - \mathbf{y}\right) + \frac{1}{2}\|\hat{\mathbf{y}}(k) - \mathbf{y}\|_2^2 \\
=&\frac{1}{2}\|\hat{\mathbf{y}}(k+1) - \hat{\mathbf{y}}(k)\|_2^2 + \left(\hat{\mathbf{y}}(k+1) - \hat{\mathbf{y}}(k)\right)^\top \left(\hat{\mathbf{y}}(k) - \mathbf{y}\right) + \mathcal{L}(\theta(k))
\end{aligned}$$

As discussed in chapter 2, smooth condition ensure that by using gradient information, GD can make a sufficient step toward a critical point while PL condition translate gradient information into distance from the optimum. Basically, smooth condition give us a upper bound on the progress $\mathcal{L}(\theta_{t+1}) - \mathcal{L}(\theta_t)$ which can be made negative by chosing a small enough step size η. Since we use ReLU activation, the smoothness condition is not straight-forward and prove that the loss function satisfies smooth condition w.r.t the network's parameters is hard since it depends on many factor. To overcome this problem, find conditions such that the loss function have a smoothness-like property. First, we decompose the progress. Let $\mathbf{g} = Z(k)^\top \mathbf{u}(k+1)$, we have

$$\begin{aligned}
&\mathcal{L}(\theta(k+1)) - \mathcal{L}(\theta(k)) \\
=&\frac{1}{2}\|\hat{\mathbf{y}}(k+1) - \hat{\mathbf{y}}(k)\|_2^2 + (\hat{\mathbf{y}}(k+1) - \mathbf{g})^\top(\hat{\mathbf{y}}(k) - \mathbf{y}) + (\mathbf{g} - \hat{\mathbf{y}}(k))^\top(\hat{\mathbf{y}}(k) - \mathbf{y}) \\
=&\underbrace{\frac{1}{2}\|\hat{\mathbf{y}}(k+1) - \hat{\mathbf{y}}(k)\|_2^2}_{I_1} + \underbrace{\mathbf{u}(k+1)^\top(Z(k+1) - Z(k))(\hat{\mathbf{y}}(k) - \mathbf{y})}_{I_2} \\
&+ \underbrace{(\mathbf{u}(k+1) - \mathbf{u}(k))^\top Z(k)(\hat{\mathbf{y}}(k) - \mathbf{y})}_{I_3}
\end{aligned}$$

This equation show if $I_1 + I_2 + I_3 < 0$, then the loss decrease. We will show the I_3 term, which is proportional to η, driving the loss function to decrease. Since $\mathbf{u}(k+1) - \mathbf{u}(k) = -\eta\nabla_{\mathbf{u}}\mathcal{L}(\theta(k))$ we observe that

$$\begin{aligned}
I_3 =& -\eta\nabla_{\mathbf{u}}\mathcal{L}(\theta(k))^\top Z(k)(\hat{\mathbf{y}}(k) - \mathbf{y}) \\
=& -\eta(\hat{\mathbf{y}}(k) - \mathbf{y})^\top Z(k)^\top Z(k)(\hat{\mathbf{y}}(k) - \mathbf{y}) \\
\leq& -\lambda_{\min}\left(Z(k)^\top Z(k)\right)\|\hat{\mathbf{y}}(k) - \mathbf{y}\|_2^2
\end{aligned}$$

For the perturbation terms I_1 and I_2, we have, by following lemma 3.3.4 if (1) the norm of the parameter is bounded then the terms I_1 is proportional to $\eta^2 C_1 \mathcal{L}(\theta(k))$ while I_2 is proportional $\eta C_2 \mathcal{L}(\theta(k))$. So if we set η sufficiently small, then I_1 is smaller than I_2 and thus (2) the condition $\lambda_{\min}\left(Z(k)^\top Z(k)\right) - C_2 \geq \frac{\lambda_0}{4}$ is sufficient for the loss function to decrease with a linear rate. Now observe that the conditions (1) and (2) can, in fact, be satisfied by large enough λ_0 which we state in condition 2.
So the proof can be decomposed into 3 steps:

1. Define a perturbation region at the initialization and prove that monDEQ attain good properties within such region $\mathcal{B}(A_0, R_A), \mathcal{B}(B_0, R_B), \mathcal{B}(U_0, R_U), \mathcal{B}(\mathbf{u}_0, R_u)$. (In fact, these good properties are direct consequences of conditions on λ_0).

2. Prove that if after k step the weights are still in this region then the parameters at step $k+1$ are also in this region. This can be proved using a path length bound which is based on the fact that the loss exponentially decreases when the parameters are in this region.

3. Prove the linear convergence rate if the parameters stay in this region (the result of good properties).

3.3.3 Proof of theorem 3.3.5

We prove the theorem by induction. First, let's define a ball around each parameters called $\mathcal{B}(A(0), R_A)$, $\mathcal{B}(B(0), R_B)$, $\mathcal{B}(U(0), R_U)$ and $\mathcal{B}(\mathbf{u}(0), R_u)$, and let $\bar{\lambda}_{\cdot} = \| \cdot_0 \|_2 + R_{\cdot}$ which is the upper bound on the operator norm of each parameters. We state induction hypotheses in the following.

Hypothesis 1. At the k-th iteration, we have $\cdot_k$ is within the ball $\mathcal{B}(\cdot_0, R_{\cdot})$

Hypothesis 2. At the k-th iteration, we have $\sigma_{\min}(Z(k)) \geq \frac{\lambda_0}{2}$.

Hypothesis 3. At the k-th iteration, we have

$$\mathcal{L}(\theta(k)) \leq \left(1 - \eta \frac{\lambda_0^2}{4}\right)^k \mathcal{L}(\theta(0)) \tag{3.20}$$

.

In the base case, where $t = 0$, it is trivial that these hypotheses 1, 2 and 3 hold.

Since the proof depend on propagation of perturbation on parameters to the output, let us state two lemma 3.3.6 and 3.3.7 which bound the perturbation in $\|W\|_2$ when A and B are perturbed and the perturbation on parameter between two consecutive step of GD, respectively.

Lemma 3.3.6 (Perturbation on $\|W\|_2$ induce by perturbation on A and B)**.** *we have*

$$\begin{aligned}\|\Delta_W\|_2 &= \left\|A^T \Delta_A + \Delta_A^T A + \Delta_A^T \Delta_A + \Delta_B - \Delta_B^T\right\|_2 \\ &\leq 2\|A\|_2 \|\Delta_A\|_2 + \|\Delta_A\|_2^2 + 2\|\Delta_B\|_2\end{aligned}$$

Lemma 3.3.7. *If hypothesis 1 holds for $k' = 1, \dots, k$, then we have*

$$
\begin{aligned}
\|U(k+1) - U(k)\|_2 =& \|\nabla_U \mathcal{L}(\theta(k))\|_F \le \eta \|X\|_F \|\hat{\mathbf{y}}(k) - \mathbf{y}\|_2 \frac{\bar{\lambda}_{\mathbf{u}}}{m} \\
\|\mathbf{u}(k+1) - \mathbf{u}(k)\|_2 =& \|\nabla_{\mathbf{u}} \mathcal{L}(\theta(k))\|_F \le \eta \|X\|_F \|\hat{\mathbf{y}}(k) - \mathbf{y}\|_2 \frac{\bar{\lambda}_U}{m} \\
\|A(k+1) - A(k)\|_2 =& \|\nabla_A \mathcal{L}(\theta(k))\|_F \le \eta \|X\|_F \|\hat{\mathbf{y}}(k) - \mathbf{y}\|_2 \frac{2\bar{\lambda}_{U\mathbf{u}A}}{m^2} \\
\|B(k+1) - B(k)\|_2 =& \|\nabla_B \mathcal{L}(\theta(k))\|_F \le \eta \|X\|_F \|\hat{\mathbf{y}}(k) - \mathbf{y}\|_2 \frac{2\bar{\lambda}_{U\mathbf{u}}}{m^2} \\
\|W(k+1) - W(k)\|_2 \le& \eta \|X\|_F \|\hat{\mathbf{y}}(k) - \mathbf{y}\|_2 \frac{4\bar{\lambda}_{U\mathbf{u}}(\bar{\lambda}_A^2 + \eta\lambda_0^2\bar{\lambda}_A^2/16 + 1)}{m^2}
\end{aligned}
$$

Proof. This lemma is a direct consequence of 3.2.3 and

$$
\|W(k+1) - W(k)\|_2 = 2\bar{\lambda_A}\|A(k+1) - A(k)\|_2 + \|A(k+1) - A(k)\|_2^2 + 2\|B(k+1) - B(k)\|_2
$$

□

Now we are able to prove that hypothesis 1 hold for step $k+1$.

Proposition 5 (Hypthesis 1). *If hypothesis 1 and 3 hold for $k' = 1, \dots, k$, we have hypothesis 1 hold for $k+1$ and*

$$
\begin{aligned}
\|A(k+1)\|_F \le \|A(0)\| + R_A = \bar{\lambda}_A \\
\|B(k+1)\|_F \le \|B(0)\| + R_B = \bar{\lambda}_B \\
\|U(k+1)\|_F \le \|U(0)\| + R_U = \bar{\lambda}_U \\
\|\mathbf{u}(k+1)\|_2 \le \|\mathbf{u}(0)\| + R_u = \bar{\lambda}_u.
\end{aligned}
$$

Further

$$
\|W(k+1) - W(0)\|_F \le \|X\|_F \|\hat{\mathbf{y}}_0 - \mathbf{y}\|_F \frac{16(3\bar{\lambda}_A^2 + 2)\bar{\lambda}_{U\mathbf{u}}}{\lambda_0^2 m^2}
$$

Proof. By triangle inequality, we have

$$
\begin{aligned}
\|A_{k+1} - A_0\|_F \le \sum_{s=0}^{k} \|A_{s+1} - A_s\|_F = \eta \sum_{s=0}^{k} &\|\nabla_A \mathcal{L}(\theta_s)\|_F \\
&\le \eta \|X\|_F \|\hat{\mathbf{y}}(k) - \mathbf{y}\|_2 \frac{2\bar{\lambda}_{U\mathbf{u}A}}{m^2} \\
&\le \eta \|X\|_F \| \frac{2\bar{\lambda}_{U\mathbf{u}A}}{m^2} \sum_{s=0}^{k} \left(1 - \eta\frac{\lambda_0^2}{4}\right)^{s/2} \|\hat{\mathbf{y}}(0) - \mathbf{y}\|_F
\end{aligned}
$$

where the second inequality follows the lemma3.3.7 and the last inequality follows by induction hypothesis 3.20. Let $u = \left(1 - \eta\frac{\alpha_0^2}{4}\right)^{1/2}$, then

$$\begin{aligned}
\|A_{k+1} - A_0\|_F &\leq \eta\frac{2\bar{\lambda}_{U\mathbf{u}A}}{m^2}\|X\|_F(1+u)\frac{1-u^{k+1}}{1-u^2}\|\hat{\mathbf{y}}_0 - \mathbf{y}\|_F \\
&\leq \frac{16\bar{\lambda}_{U\mathbf{u}A}}{m^2}\|X\|_F\|\hat{\mathbf{y}}_0 - \mathbf{y}\|_F \\
&\leq R_A
\end{aligned}$$

where the last inequality follows from condition 3.16. By triangle inequality, this implies

$$\|A(k+1)\|_2 \leq \|A_0\| + R_A = \bar{\lambda}_A \tag{3.21}$$

Similarly we have

$$\begin{aligned}
\|B(k+1) - B(0)\|_F \leq& \frac{16}{\lambda_0^2}\frac{1}{m^2}\|X\|_F\bar{\lambda}_U\bar{\lambda}_u\|\hat{\mathbf{y}}(0) - \mathbf{y}\|_F \leq R_B \\
\|U(k+1) - U(0)\|_F \leq& \frac{8}{\lambda_0^2}\frac{1}{m}\|X\|_F\bar{\lambda}_u\|\hat{\mathbf{y}}(0) - \mathbf{y}\|_F \leq R_U \\
\|\mathbf{u}(k+1) - \mathbf{u}(0)\|_F \leq& \frac{8}{\lambda_0^2}\frac{1}{m}\|X\|_F\bar{\lambda}_u\|\hat{\mathbf{y}}(0) - \mathbf{y}\|_F \leq R_{\mathbf{u}}
\end{aligned}$$

So the hypothesis 1 hold for $k+1$ and we also have

$$\begin{aligned}
\|B(k+1)\|_2 &\leq \|B(0)\| + R_B = \bar{\lambda}_B \\
\|U(k+1)\|_2 &\leq \|U(0)\| + R_U = \bar{\lambda}_U \\
\|\mathbf{u}(k+1)\|_2 &\leq \|\mathbf{u}(0)\| + R_u = \bar{\lambda}_u
\end{aligned}$$

For the bound on $\|W(k+1) - W(0)\|_F$, we use lemma 3.3.6 and have

$$\begin{aligned}
&\|W(k+1) - W(0)\|_F \\
\leq&\|A(k+1) - A(0)\|_F^2 + 2\|A(k+1) - A(0)\|_F\|A(0)\|_2 + 2\|B(k+1) - B(0)\| \\
\leq&\frac{256}{\lambda_0^4 m^4}\bar{\lambda}_{U\mathbf{u}A}^2\|\hat{\mathbf{y}}_0 - \mathbf{y}\|_F^2\|X\|_F + \frac{32}{\lambda_0^2 m^2}\bar{\lambda}_{U\mathbf{u}A}\|\hat{\mathbf{y}}_0 - \mathbf{y}\|_F\|X\|_F^2\bar{\lambda}_A + \frac{32}{\lambda_0^2 m^2}\bar{\lambda}_{U\mathbf{u}}\|\hat{\mathbf{y}}_0 - \mathbf{y}\|_F\|X\|_F^2 \\
\leq&\frac{16}{\lambda_0^2 m^2}\bar{\lambda}_{U\mathbf{u}}\bar{\lambda}_A^2\|\hat{\mathbf{y}}_0 - \mathbf{y}\|_F\|X\|_F + \frac{32}{\lambda_0^2 m^2}\bar{\lambda}_{U\mathbf{u}A}\|\hat{\mathbf{y}}_0 - \mathbf{y}\|_F\|X\|_F\bar{\lambda}_A + \frac{32}{\lambda_0^2 m^2}\bar{\lambda}_{U\mathbf{u}}\|\hat{\mathbf{y}}_0 - \mathbf{y}\|_F\|X\|_F \\
\leq&\|\hat{\mathbf{y}}_0 - \mathbf{y}\|_F\|X\|_F\frac{16(3\bar{\lambda}_A^2 + 2)\bar{\lambda}_{U\mathbf{u}}}{\lambda_0^2 m^2}
\end{aligned}$$

□

Proposition 6 (Hypothesis 2). *If hypothesis 1 and 3 hold for $k' = 1, \dots, k$, we have hypothesis 2 hold for $k+1$ i.e*

$$\sigma_{\min}(Z(k+1)) \geq \frac{\lambda_0}{2}$$

Proof.

$$\begin{aligned}
&\|Z_{k+1} - Z_0\| \\
\leq &\|X\|_F \left(\frac{\|U_{k+1} - U_0\|_2}{m} + \frac{\|W_{k+1} - W_0\|_2 \|U_0\|_2}{m^2} \right) \\
\leq &\|X\|_F \left(\|X\|_F \|\hat{\mathbf{y}}(0) - \mathbf{y}\|_2 \frac{8\bar{\lambda}_{\mathbf{u}}}{\lambda_0^2} + \|X\|_F \|\hat{\mathbf{y}}_0 - \mathbf{y}\|_2 \frac{16(3\bar{\lambda}_A^2 + 2)\bar{\lambda}_{U\mathbf{u}}}{\lambda_0^2} \bar{\lambda}_U \right) \\
\leq &\|X\|_F^2 \|\hat{\mathbf{y}}(0) - \mathbf{y}\|_2 \frac{8\bar{\lambda}_{\mathbf{u}} + 16(3\bar{\lambda}_A^2 + 2)\bar{\lambda}_U^2 \bar{\lambda}_{\mathbf{u}}}{\lambda_0^2} \\
\leq &\frac{\lambda_0}{2}
\end{aligned}$$

where the last inequality follows from condition 3.17. By Weyl's inequality, we have

$$\sigma_{\min}(Z(k+1)) \geq \sigma_{\min}(Z(0)) - \|Z(k+1) - Z(0)\| \geq \frac{\lambda_0}{2}$$

□

Now, we follow the proof sketch described in previous part. We first analyze the perturbation terms I_1 and I_2.

Lemma 3.3.8. *If hypothesis 1 and 3 hold for $k' = 1, \dots, k$, we have*

$$\begin{aligned}
I_1 = &\frac{1}{2} \|\hat{\mathbf{y}}_{k+1} - \hat{\mathbf{y}}_k\|_2^2 \\
\leq &\mathcal{L}(\theta_k) \left[\eta^2 C_{11}^2 + 2\eta^3 \lambda_0^2 C_{11} C_{12} + \eta^4 \lambda_0 C_{12}^2 \right]
\end{aligned}$$

where $C_{11} = \|X\|_F^2 (\bar{\lambda}_u^2 + 4\bar{\lambda}_{Uu}^2 + 4\bar{\lambda}_{UuA}^2 + \bar{\lambda}_U^2)$ and $C_{12} = \|X\|_F^2 \bar{\lambda}_{UuA}^2 / 4$.

Proof. We use lemma 3.3.4 and lemma 3.3.7 which is like a Lipschitz condition of the predictions and have

$$\begin{aligned}
&\|\hat{\mathbf{y}}(k+1) - \hat{\mathbf{y}}(k)\|_2 \\
\leq &\|X\|_F \left(\frac{\bar{\lambda}_u \|U(k+1) - U(k)\|_2}{m} + \frac{\bar{\lambda}_u \|W(k+1) - W(k)\|_2 \|U(k)\|_2}{m^2} + \frac{\bar{\lambda}_U \|\mathbf{u}(k+1) - \mathbf{u}(k)\|_2}{m} \right) \\
\leq &\|X\|_F^2 \|\hat{\mathbf{y}}(k) - \mathbf{y}\|_2 \eta \left(\bar{\lambda}_u^2 + 4\bar{\lambda}_{U\mathbf{u}A}^2 + \eta \lambda_0^2 \bar{\lambda}_{U\mathbf{u}A}^2 / 4 + 4\bar{\lambda}_{U\mathbf{u}}^2 + \bar{\lambda}_U^2 \right) \\
\leq &\|X\|_F^2 \|\hat{\mathbf{y}}(k) - \mathbf{y}\|_2 \left[\eta \left(\bar{\lambda}_u^2 + 4\bar{\lambda}_{U\mathbf{u}A}^2 + 4\bar{\lambda}_{U\mathbf{u}}^2 + \bar{\lambda}_U^2 \right) + \eta^2 \lambda_0^2 \bar{\lambda}_{U\mathbf{u}A}^2 / 4 \right] \\
= &\sqrt{2\mathcal{L}(\theta(k))} \left[\eta C_{11} + \eta^2 \lambda_0^2 C_{12} \right]
\end{aligned}$$

So

$$\begin{aligned}&\frac{1}{2}\|\hat{\mathbf{y}}_{k+1}-\hat{\mathbf{y}}_k\|_2^2\\ \leq&\mathcal{L}(\theta_k)\left[\eta^2C_{11}^2+2\eta^3\lambda_0^2C_{11}C_{12}+\eta^4\lambda_0C_{12}^2\right]\end{aligned}$$

□

Lemma 3.3.9. *If hypothesis 1 and 3 hold for $k'=1,\dots,k$, we have*

$$\begin{aligned}I_2=&(\hat{\mathbf{y}}_{k+1}-\mathbf{g})^\top(\hat{\mathbf{y}}_k-\mathbf{y})\\ \leq&2\mathcal{L}(\theta(k))\left[\eta(C_{11}-\bar{\lambda}_U^2\|X\|_F^2)+\eta^2\lambda_0^2C_{12}\right]\end{aligned}$$

Proof.

$$\begin{aligned}&(\hat{\mathbf{y}}_{k+1}-\mathbf{g})^\top(\hat{\mathbf{y}}_k-\mathbf{y})\\ \leq&\|Z_{k+1}-Z_k\|_F\|\mathbf{u_{k+1}}\|_2\|\hat{\mathbf{y}}_k-\mathbf{y}\|\\ \leq&\left(\frac{\bar{\lambda}_u\|U_{k+1}-U_k\|_2}{m}+\frac{\bar{\lambda}_u\|W_{k+1}-W_k\|_2\|U_k\|_2}{m^2}\right)\|X\|_F\|\hat{\mathbf{y}}_k-\mathbf{y}\|_2\\ \leq&\left(\frac{\bar{\lambda}_u\|U_{k+1}-U_k\|_2}{m}+\frac{\bar{\lambda}_u\bar{\lambda}_U\|W_{k+1}-W_k\|_2}{m^2}\right)\|X\|_F\|\hat{\mathbf{y}}_k-\mathbf{y}\|_2\\ \leq&\|X\|_F^2\|\hat{\mathbf{y}}(k)-\mathbf{y}\|_2^2\left[\eta\left(\bar{\lambda}_u^2+4\bar{\lambda}_{U\mathbf{u}A}^2+4\bar{\lambda}_{U\mathbf{u}}^2\right)+\eta^2\lambda_0^2\bar{\lambda}_{U\mathbf{u}A}^2/4\right]\\ =&2\mathcal{L}(\theta(k))\left[\eta(C_{11}-\bar{\lambda}_U^2\|X\|_F^2)+\eta^2\lambda_0^2C_{12}\right]\end{aligned}$$

□

Lemma 3.3.10. *If $n\geq N$ and hypothesis 1, 2 and 3 hold for $k'=1,\dots,k$, we have*

$$\begin{aligned}I_3=&(\mathbf{g}-\hat{\mathbf{y}}_k)^\top(\hat{\mathbf{y}}_k-\mathbf{y})\\ \leq&-\eta\frac{\lambda_0^2}{2}\mathcal{L}(\theta_k)\end{aligned}$$

Proof. Using the fact $(\mathbf{u}_{k+1}-\mathbf{u}_k)=-\eta\nabla_{\mathbf{u}}\mathcal{L}(k)$, we have

$$\begin{aligned}&(\mathbf{g}-\hat{\mathbf{y}}_k)^\top(\hat{\mathbf{y}}_k-\mathbf{y})\\ =&-\eta\nabla_{\mathbf{u}}\mathcal{L}(k)Z_k^\top(\hat{\mathbf{y}}_k-\mathbf{y})\\ =&-\eta(\hat{\mathbf{y}}_k-\mathbf{y})^\top Z_kZ_k^\top(\hat{\mathbf{y}}_k-\mathbf{y})\\ \leq&-\eta\sigma_{\min}(Z_k)^2\|\hat{\mathbf{y}}_k-\mathbf{y}\|_2^2\\ \leq&-\eta2\sigma_{\min}(Z_k)^2\mathcal{L}(\theta_k)\\ \leq&-\eta2\left(\sigma_{\min}(Z_0)-\|Z_k-Z_0\|_F\right)^2\mathcal{L}(\theta_k),\quad\text{by Weyl's inequality}\\ \leq&-\eta\frac{\lambda_0^2}{2}\mathcal{L}(\theta_k)\end{aligned}$$

where we need our assumption $n \geq N$ to optain $\lambda_{\min}(ZZ^\top) = \sigma^2_{\min}(Z)$ and the last inequality follows from hypothesis 2. $\square$

Putting all bounds together, we have

$$\begin{aligned}
\mathcal{L}(k+1) \leq & \mathcal{L}(k) + I_1 + I_2 + I_3 \\
\leq & \mathcal{L}(k) + \mathcal{L}(\theta_k)\eta^2 \left[C_{11} + \eta\lambda_0^2 C_{12}\right]^2 + \mathcal{L}(\theta_k)\eta 2 \left[(C_{11} - \bar{\lambda}_U^2) + \eta\lambda_0^2 C_{12}\right] - \eta\frac{\lambda_0^2}{2}\mathcal{L}(\theta_k) \\
\leq & \mathcal{L}(\theta_k) \left[1 + \eta^2 C_{11}^2 + 2\eta^3\lambda_0^2 C_{11}C_{12} + \eta^4\lambda_0 C_{12}^2 + 2\eta(C_{11} - \bar{\lambda}_U^2\|X\|_F^2) + 2\eta^2\lambda_0^2 C_{12} - \eta\frac{\lambda_0^2}{2}\right] \\
\leq & \mathcal{L}(\theta_k) \left[1 + 5\eta C_{11} - 2\eta\bar{\lambda}_U^2\|X\|_F^2 + 3\eta^2\lambda_0^2 C_{12} - \eta\frac{\lambda_0^2}{2}\right] \\
\leq & \mathcal{L}(\theta_k) \left[1 + 5\eta C_{11} - \eta\frac{\lambda_0^2}{2}\right] \quad \text{by condition on } \eta \text{ s.t } 3\eta\lambda_0^2 C_{12} \leq 2\bar{\lambda}_U^2\|X\|_F^2 \\
\leq & \mathcal{L}(\theta_k) \left[1 - \eta\frac{\lambda_0^2}{4}\right] \quad \text{by condition on } \lambda_0 \text{ s.t } 5C_{11} \leq \frac{\lambda_0^2}{4}
\end{aligned}$$

where the forth and fifth inequality follows from condition on η and the last inequality follows from condition 3.18 on λ_0.

Chapter 4

Experiments

4.1 Experiments on the dynamics

In this section, we will describe some experiments on the forward dynamic of DEQ models. We focus on a vision task and use a simple variant of residual network architecture, which can be written as

$$f(\mathbf{z}, \mathbf{x}) = \text{norm}(\text{ReLU}(\mathbf{z} + \text{norm}(\mathbf{x} + W_2 * (\text{norm}(\text{ReLU}(W_1 * \mathbf{z})))))) \tag{4.1}$$

where the norm layers are group norm (Wu and He, 2018) and $*$ denotes the convolution operator. To evaluate how close is an equilibrium point proposal to the actual fixed point, we use the average (relative) residual over all the batches which are defined as

$$\frac{\|f(\mathbf{z}^{[k]}, \mathbf{x}) - \mathbf{z}^{[k]}\|_2}{\|f(\mathbf{z}^{[k]}, \mathbf{x})\|_2}. \tag{4.2}$$

For the fixed point solver, we choose the Anderson Acceleration method.

4.1.1 Initialization

The choice of architecture and the initialization of the weight of DEQ might affect the fixed point convergence rate. One oddity of DEQ models is that the weights of these convolutional operators typically need to be initialized with smaller values than layers in traditional networks. To demonstrate this behavior, we compare the convergence of $\mathbf{z}^{[i]}$ with two different initializations, one from standard $\mathcal{N}(0, 1)$ and the other from a smaller variance normal distribution $\mathcal{N}(0, 0.01)$. We plot the (relative) residuals over difference input which is sample from MNIST test set (Fig. 4.1). We also plot the convergent trajectory in these two settings by using PCA to reduce the number of dimension of hidden vector $\mathbf{z}^{[i]}$ to 2 (Fig. 4.2).

As can be seen from initial experiments (Fig. 4.1 and Fig 4.2), the sequence of $\mathbf{z}^{[i]}$ is more chaos in the normal initialization setting while in small initialization, $\mathbf{z}^{[i]}$ stay quite steady after the first step. The precise ideal scaling laws for the variance terms are still not

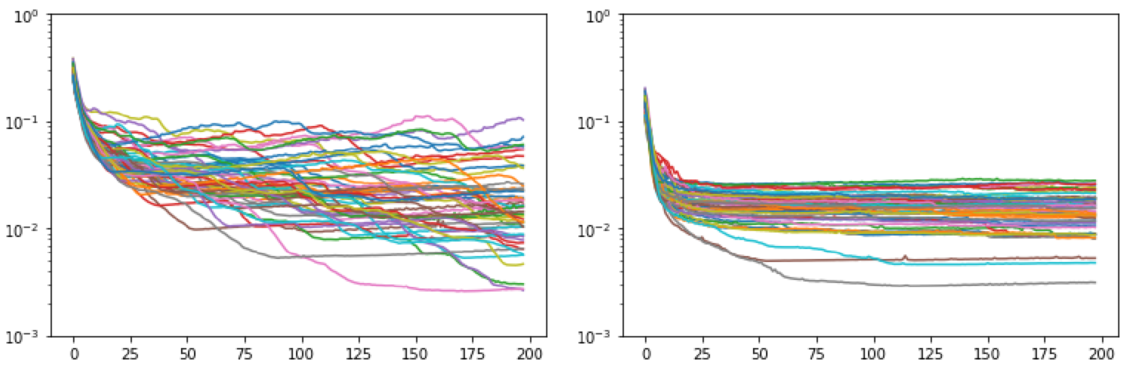


Figure 4.1: The convergence of $\mathbf{z}^{[i]}$ with inputs from MNIST dataset. On the left is the convergence of the hidden vector with standard $\mathcal{N}(0, 1)$ initialization. On the right is the convergence of hidden vector with small initialization $\mathcal{N}(0, 0.01)$.

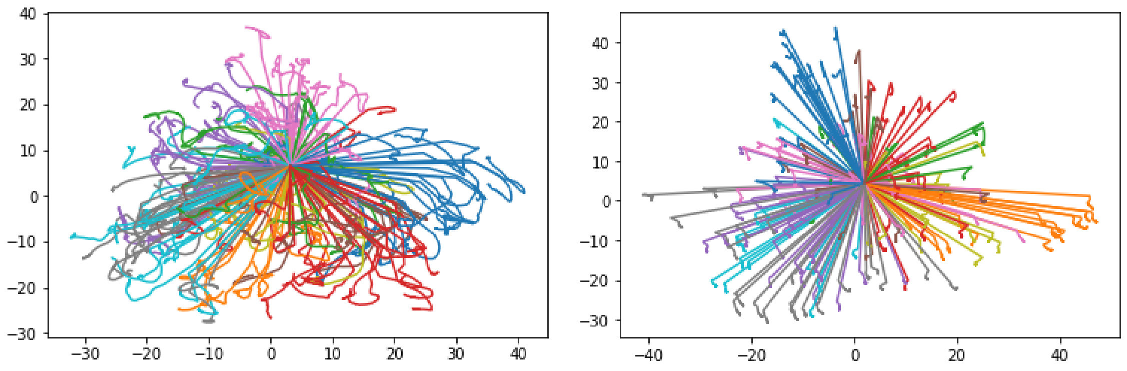


Figure 4.2: PCA projection of the trajectory of $\mathbf{z}^{[i]}$ with inputs from MNIST dataset. On the left is the trajectory of hidden vectors with standard $\mathcal{N}(0, 1)$ initialization. On the right is the trajectory of hidden vectors with small initialization $\mathcal{N}(0, 0.01)$.

well understood (though a reasonably wide range of this smaller variance initialization works well) and might result from the connection of the spectral radius of J_f and the stability of the networks as discussed in 2.3.

4.1.2 Change of the dynamic during training

The function f of the DEQ layer can be seen as an evolutionary function and the sequence of $\mathbf{z}^{[i]}$ produced by the solver can be interpreted as a trajectory of f starting at $\mathbf{z}^{[0]}$. Note that the evolution of $\mathbf{z}^{[i]}$ is dependent on the input $\mathbf{x}$ by input injection.

In this experiment, we train the previous model on MNIST datasets and analyze the change of the dynamic f of DEQ during training. $\mathbf{z}^{[0]}$ is initialized to a zero vector so all the trajectories start from the origin. We train the model with 40 epochs, using Adam optimizer with the learning rate 0.001 and with CosineAnnealingLR scheduler. We use the Anderson solver with the tolerance is $1e-2$ and maximum of 50 iterations. The trajectory is sampled by feeding the inputs from the training dataset and collecting the intermediate states from steps of the solver. By using PCA to reduce the number dimension of $\mathbf{z}^{[i]}$ to 2, we are able to plot these trajectories in a plane (Fig. 4.3).

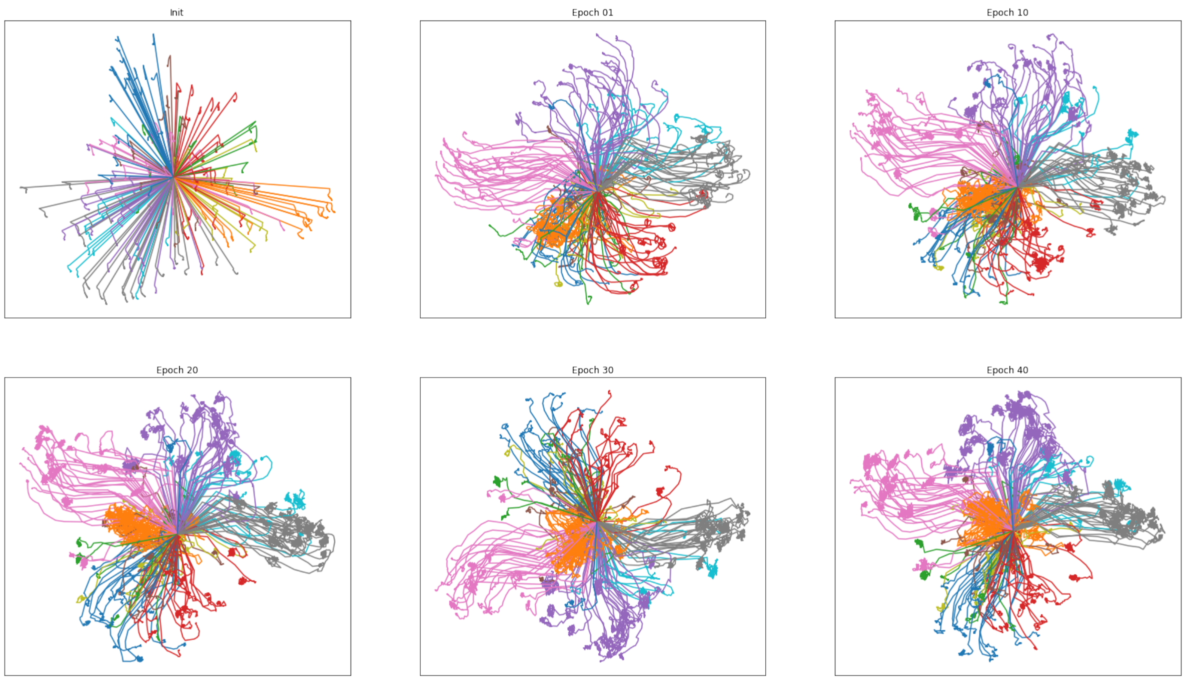
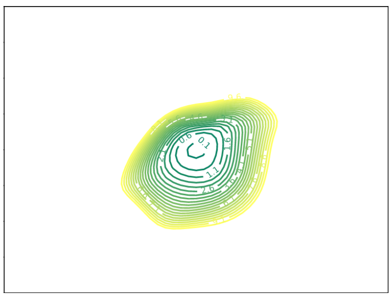


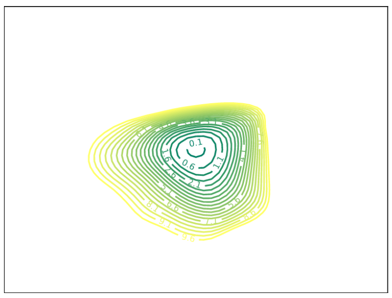
Figure 4.3: The change of trajectory of $\mathbf{z}^{[i]}$ with inputs from MNIST train dataset during training using AA solver with $tol = 1e-3$ and maximum 100 iterations. Difference colors represent different classes.

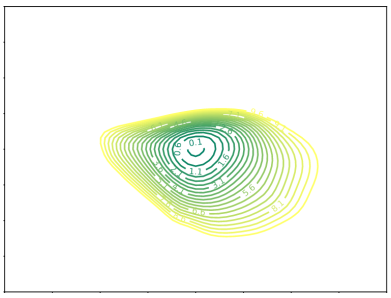
We obtain the same observation with Bai et al. (2021), the dynamic of DEQ becomes more complex during training, which can be interpreted as the model becoming deeper and more complex. In initialization, the trajectory of h just takes a single leap and quickly becomes steady. After the first epoch, one can see from Fig. 4.3 that the hidden states of DEQ corresponding to different classes are moved to a different region of the space and it seems to be converged to a fixed point as it reaches that region. During the training, the trajectory becomes more complex and orbits near the fixed points.

4.2 Visualizing loss landscape

We plot the loss surface of a DEQ model using a range of visualization methods. While visualizations are only possible using at most 2D (surface) plots, neural networks typically contain many parameters (even with weight tying), so their loss functions live in a very high-dimensional space. For closing this dimensionality gap, the "random directions" approach can be used. By choosing a center point θ^* in the graph and choosing two direction vectors d_1 and d_2, one then be able to plot a function of the form $\Phi(\alpha, \beta) = \mathcal{L}(\theta^* + \alpha d_1 + \beta d_2)$ in the 2D (surface) case. In our experiment, the interested point θ^* is the minima found by


(a) repeat-6 net


(b) repeat-11 net


(c) repeat-18 net

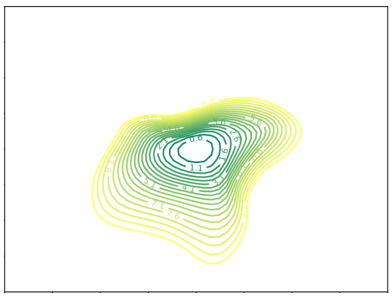

(d) DEQ net

Figure 4.4: Contour plot of loss surface of trained models around minimizers

the optimizer. The main focus of us in this experiment is to explore the geometry of the minima, especially the flatness property. As Dinh et al. (2017) show, flatness is sensitive to reparametrization such as node-wise re-scalings of ReLU nets or general coordinate transformations. Although the "random directions" approach to plotting is simple, it fails to capture the intrinsic geometry of loss surfaces, and cannot be used to compare the geometry of two different minimizers or two different networks. To address this problem, we use a technique called filter-wise normalized directions which is proposed by Li et al. (2017). This method is invariant to the particular type of reparametrizations considered by Dinh et al. (2017).

4.2.1 Comparison with explicit weight-tying models

For comparison, we use three weight-tying models with the same architecture with DEQ model in the above section but replace the DEQ layer with a weight-tying layer, which repeats the function f_θ on $\mathbf{z}$ and $\mathbf{x}$. Each of these models will repeat 6, 11 and 18 times, respectively. We call these networks repeat-6,11,18.

all models are train on the CIFAR10 dataset using ADAM with Cosine Annealing scheduler, batch-size 100 for 50 epoch. The learning rate was initialized at 0.001.

After having the optimized weights, we plot loss surface around the minimums for each model. The ranges are from -1 to 1 along two chosen directions and the resolution is 51×51.

2D plots of the minimizers for the experimental models are shown as contour plots in Figure 4.4. Note that the center of each plot corresponds to the minimizer, and the two axes parameterize two random directions with filter-wise normalization. From the plots, we can see that the small-loss neighborhood around the minimizer is expanded as we increase the number of repetitions of the f function. This suggests the superior of training DEQ models.

4.2.2 Loss landscape of MDEQ model

We also visualized the loss landscape of a large DEQ model, particularly an MDEQ model. This model is trained on ImageNet dataset (Deng et al., 2009) which is a large dataset for image classification. Due to computational reasons, we only use 1024 examples of ImageNet test set, this set contains about 50 images for each of 20 classes. The result is plot in Figure 4.5.

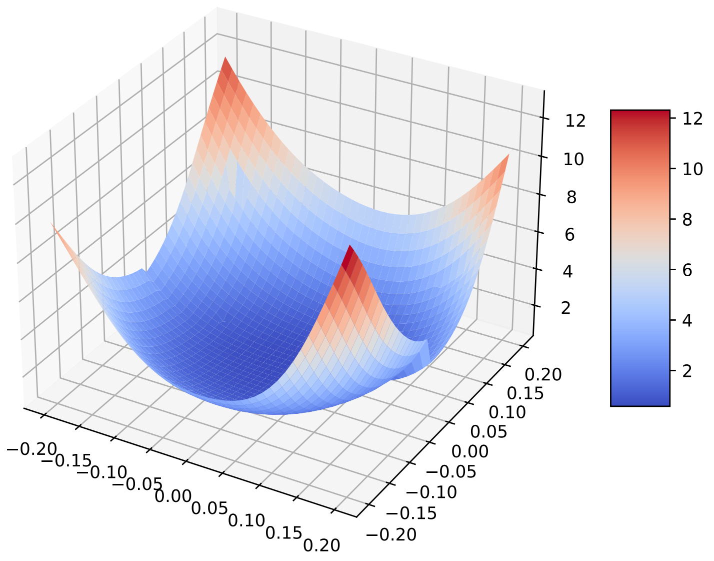


Figure 4.5: Loss landscape of trained MDEQ model on 1024 examples from ImageNet test dataset.

Chapter 5

Conclusion and Future works

In this work, we have studied training the deep equilibrium models, particularly monotone DEQ models with ReLU activation function, and show that (1) in the gradient flow dynamic, despite the "infinite" depth, non-smooth, and non-convexity, the linear convergence rate to a global optimum is guaranteed if the network satisfies suitable conditions at initialization. And (2) gradient descent with a fixed step size converges to a global minimum of implicit neural networks at a linear rate as long as the same assumptions are satisfied and the step size is small enough. Our experiment emphasized the growing complexity of the forward pass of deep equilibrium models. Furthermore, our visualizations of the loss landscape of the MDEQ model, which is a large-scale DEQ model on ImageNet, show that this model might have only a minimum and an invex-like loss landscape which indicates that training these models might attain a linear convergence rate as our analysis suggest.
In future work,

- We are interested in provides concrete discussion on the satisfiability of initial conditions on λ_0 in theorem 3.3.1 and 3.3.5 under different initialization schemes.
- In our visualization of the loss landscape of DEQ models, we also observe that the minimum is flat, suggesting that the DEQ model might have some promising properties in terms of generalization performance. With this observation, we are interested in using our framework to analyze the implicit bias of implicit layer models.

References

ALMEIDA, L. B. (1987). A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. *Proceedings, 1st First International Conference on Neural Networks*, 2:609–618.

Amos, B. and Kolter, J. Z. (2017). Optnet: Differentiable optimization as a layer in neural networks. *CoRR*, abs/1703.00443.

Anderson, D. G. (1965). Iterative procedures for nonlinear integral equations. *J. ACM*, 12(4):547–560.

Bai, S., Kolter, J. Z., and Koltun, V. (2019a). Deep equilibrium models. *CoRR*, abs/1909.01377.

Bai, S., Kolter, J. Z., and Koltun, V. (2019b). Trellis networks for sequence modeling. *ArXiv*, abs/1810.06682.

Bai, S., Koltun, V., and Kolter, J. Z. (2020). Multiscale deep equilibrium models. *CoRR*, abs/2006.08656.

Bai, S., Koltun, V., and Kolter, J. Z. (2021). Stabilizing equilibrium models by jacobian regularization. *CoRR*, abs/2106.14342.

Bai, S., Koltun, V., and Kolter, J. Z. (2022). Neural deep equilibrium solvers. In *International Conference on Learning Representations*.

Bolte, J., Le, T., Pauwels, E., and Silveti-Falls, A. (2021). Nonsmooth implicit differentiation for machine learning and optimization. *CoRR*, abs/2106.04350.

Bolte, J. and Pauwels, E. (2021). Conservative set valued fields, automatic differentiation, stochastic gradient method and deep learning. *ArXiv*, abs/1909.10300.

Brouwer, E. D., Simm, J., Arany, A., and Moreau, Y. (2019). Gru-ode-bayes: Continuous modeling of sporadically-observed time series. *CoRR*, abs/1905.12374.

Broyden, C. G. (1965). A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation*, 19:577–593.

Chen, R. 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*, volume 31. Curran Associates, Inc.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Clarke, F. H. (1976). On the inverse function theorem. *Pacific Journal of Mathematics*, 64:97–102.

Csordás, R., Irie, K., and Schmidhuber, J. (2021). The devil is in the detail: Simple tricks improve systematic generalization of transformers. *CoRR*, abs/2108.12284.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, L. (2019). Universal transformers. *ArXiv*, abs/1807.03819.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. (2017). Sharp minima can generalize for deep nets. *CoRR*, abs/1703.04933.

Djolonga, J. and Krause, A. (2017). Differentiable learning of submodular models. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Duvenaud, D., Kolter, J. Z., and Johnson., M. (2020). Deep implicit layers tutorial - neural odes, deep equilibirum models, and beyond. *Neural Information Processing Systems Tutorial 2020*.

Falorsi, L. and Forré, P. (2020). Neural ordinary differential equations on manifolds.

Fang, H.-r. and Saad, Y. (2009). Two classes of multisecant methods for nonlinear acceleration. *Numerical Linear Algebra with Applications*, 16(3):197–221.

Gao, T. and Gao, H. (2022). Gradient descent optimizes infinite-depth relu implicit networks with linear widths. *ArXiv*, abs/2205.07463.

Ge, R., Huang, F., Jin, C., and Yuan, Y. (2015). Escaping from saddle points - online stochastic gradient for tensor decomposition. *CoRR*, abs/1503.02101.

Gilton, D., Ongie, G., and Willett, R. (2021). Deep equilibrium architectures for inverse problems in imaging. *IEEE Transactions on Computational Imaging*, 7:1123–1133.

Gupta, C., Balakrishnan, S., and Ramdas, A. (2019). Path length bounds for gradient descent and flow. *CoRR*, abs/1908.01089.

Hanin, B. (2019). Universal function approximation by deep neural nets with bounded width and relu activations. *Mathematics*, 7(10).

Hanin, B. and Sellke, M. (2017). Approximating continuous functions by relu nets of minimal width. *ArXiv*, abs/1710.11278.

He, K., Zhang, X., Ren, S., and Sun, J. (2015a). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

He, K., Zhang, X., Ren, S., and Sun, J. (2015b). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Heaton, H., Fung, S. W., Gibali, A., and Yin, W. (2021). Feasibility-based fixed point networks. *CoRR*, abs/2104.14090.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366.

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

Jin, C., Ge, R., Netrapalli, P., Kakade, S. M., and Jordan, M. I. (2017). How to escape saddle points efficiently. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1724–1732. JMLR.org.

Kawaguchi, K. (2016). Deep learning without poor local minima. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Kawaguchi, K. (2021). On the theory of implicit deep learning: Global convergence with implicit layers. *CoRR*, abs/2102.07346.

Kawaguchi, K., Kaelbling, L., and Bengio, Y. (2017). Generalization in deep learning.

Kidger, P. and Lyons, T. (2020). Universal Approximation with Deep Narrow Networks. In Abernethy, J. and Agarwal, S., editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 2306–2327. PMLR.

Kidger, P., Morrill, J., Foster, J., and Lyons, T. J. (2020). Neural controlled differential equations for irregular time series. *CoRR*, abs/2005.08926.

Kim, J., Linsley, D., Thakkar, K., and Serre, T. (2019). Disentangling neural mechanisms for perceptual grouping. *CoRR*, abs/1906.01558.

Lee, J. D., Simchowitz, M., Jordan, M. I., and Recht, B. (2016). Gradient descent converges to minimizers.

Li, H., Xu, Z., Taylor, G., and Goldstein, T. (2017). Visualizing the loss landscape of neural nets. *CoRR*, abs/1712.09913.

Liao, Q. and Poggio, T. A. (2016). Bridging the gaps between residual learning, recurrent neural networks and visual cortex. *CoRR*, abs/1604.03640.

Liao, R., Xiong, Y., Fetaya, E., Zhang, L., Yoon, K., Pitkow, X., Urtasun, R., and Zemel, R. S. (2018). Reviving and improving recurrent back-propagation. *CoRR*, abs/1803.06396.

Lindsay, G. W. (2021). Convolutional neural networks as a model of the visual system: Past, present, and future. *Journal of Cognitive Neuroscience*, 33:2017–2031.

Ling, Z., Xie, X., Wang, Q., Zhang, Z., and Lin, Z. (2022). Global convergence of over-parameterized deep equilibrium models. *ArXiv*, abs/2205.13814.

Linsley, D., Ashok, A. K., Govindarajan, L. N., Liu, R., and Serre, T. (2020). Stable and expressive recurrent vision models. *CoRR*, abs/2005.11362.

Linsley, D., Kim, J., Veerabadran, V., and Serre, T. (2018). Learning long-range spatial dependencies with horizontal gated-recurrent units. *CoRR*, abs/1805.08315.

Liu, C., Zhu, L., and Belkin, M. (2020). Toward a theory of optimization for over-parameterized systems of non-linear equations: the lessons of deep learning. *CoRR*, abs/2003.00307.

Lou, A., Lim, D., Katsman, I., Huang, L., Jiang, Q., Lim, S.-N., and De Sa, C. (2020). Neural manifold ordinary differential equations. NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

Maclaurin, D., Duvenaud, D., and Adams, R. P. (2015). Gradient-based hyperparameter optimization through reversible learning. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 2113–2122. JMLR.org.

Mathieu, E. and Nickel, M. (2020). Riemannian continuous normalizing flows. NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

Nayebi, A., Bear, D., Kubilius, J., Kar, K., Ganguli, S., Sussillo, D., DiCarlo, J. J., and Yamins, D. L. (2018). Task-driven convolutional recurrent models of the visual system. 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.

Pabbaraju, C., Winston, E., and Kolter, J. Z. (2021). Estimating lipschitz constants of monotone deep equilibrium models. In *International Conference on Learning Representations*.

Pineda, F. J. (1987). Generalization of back-propagation to recurrent neural networks. *Phys. Rev. Lett.*, 59:2229–2232.

Rubanova, Y., Chen, R. T. Q., and Duvenaud, D. (2019). Latent odes for irregularly-sampled time series. *CoRR*, abs/1907.03907.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Schwarzschild, A., Gupta, A., Goldblum, M., and Goldstein, T. (2021). Thinking deeply with recurrence: Generalizing from easy to hard sequential reasoning problems. *CoRR*, abs/2102.11011.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021). Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

Wang, P., Donti, P. L., Wilder, B., and Kolter, J. Z. (2019). Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. *CoRR*, abs/1905.12149.

Winston, E. and Kolter, J. Z. (2020). Monotone operator equilibrium networks. *CoRR*, abs/2006.08591.

Wu, Y. and He, K. (2018). Group normalization. *CoRR*, abs/1803.08494.

Yang, G., Huang, X., Hao, Z., Liu, M.-Y., Belongie, S., and Hariharan, B. (2019). Pointflow: 3d point cloud generation with continuous normalizing flows. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4540–4549.

Yun, C., Sra, S., and Jadbabaie, A. (2018). A critical view of global optimality in deep learning.

Appendix A

The Implicit Function Theorem

Consider the smooth function $g : \mathbb{R}^{d_z} \times \mathbb{R}^{d_p} \mapsto \mathbb{R}^{d_z}$ and the equation $g(z, p) = 0$. Implicit function theorem seek to address two questions: (1) Can we find a funtion $z = s(p)$ which solve our equation (i.e $g(s(p), p) = 0$) at least locally and (2) can we compute the gradient of the implicit function h. The classical implicit function theorem due to Dini and Cauchy answers both question positively.

Theorem A.0.1. *Let $g : \mathbb{R}^{d_z} \times \mathbb{R}^{d_p} \mapsto \mathbb{R}^{d_z}$ be continuously differentiable in a neighborhood of $(\bar{z}, \bar{p})$ and such that*

1. *$g(\bar{z}, \bar{p}) = 0$, and*
2. *let the partial Jacobian of g w.r.t x at $(\bar{z}, \bar{p})$, namely $\nabla_z g(\bar{z}, \bar{p})$, be nonsingular.*

Then there exist open sets $S_{\bar{p}} \subset \mathbb{R}^{d_p}$ and $S_{\bar{z}} \subset \mathbb{R}^{d_z}$ containing $\bar{p}$ and $\bar{z}$, repectively, and a unique continuous function $s : S_{\bar{p}} \mapsto S_{\bar{z}}$ such that:

1. *$\bar{z} = s(\bar{p})$,*
2. *$g(s(p), p) = 0 \; \forall p \in S_{\bar{p}}$, and*
3. *s is differentiable on $S_{\bar{p}}$*

$$\nabla s(p) = -\nabla_z g(s(p), p)^{-1} \nabla_p g(s(p), p) \text{ for every } p \in S_{\bar{p}}. \tag{A.1}$$

The hypothesis of the theorem are that the function g is somehow nice (continuously differentiable) and consider a point at which the function g attains a zero. The first part of implicit function theorem said that as long as we avoid "bad" points, for which the Jacobian of g w.r.t z is singular, then there is a neighborhood $S_{\bar{p}}$ on which there exist a smooth implicit function s which solve the equation $g(z, p) = 0$. This answer the question of existence. Then the theorem goes on to state how can we compute the gradient of the implicit function s in term of the gradient of g. This provides us with the caculus for the gradient of the implicit function s.

What happend in the non-smooth setting? The question of existence is addressed positively by Clarke (1976) for locally Lipschitz functions. Recently, Bolte et al. (2021) answer the question of how to compute the corresponding gradient in this setting by using convervative Jacobian (Bolte and Pauwels, 2021) and path differentiable functions.

Appendix B

Motivation - Weight-tied Network

The motivation of DEQ starts from works that employ the same transformation in each layer (known as weight tying) and still achieve results competitive with the state-of-the-art. For instance, Trellis networks (Bai et al., 2019b) introduced a state-of-the-art architecture which is both a feedforward convolutional neural network and a truncated RNN with an LSTM cell at the core. Another example is Universal transformers (Dehghani et al., 2019) which benefit from the recurrent bias of RNNs, obtaining a feedforward architecture with dynamic depth, which is proved to be Turing complete under some assumptions.

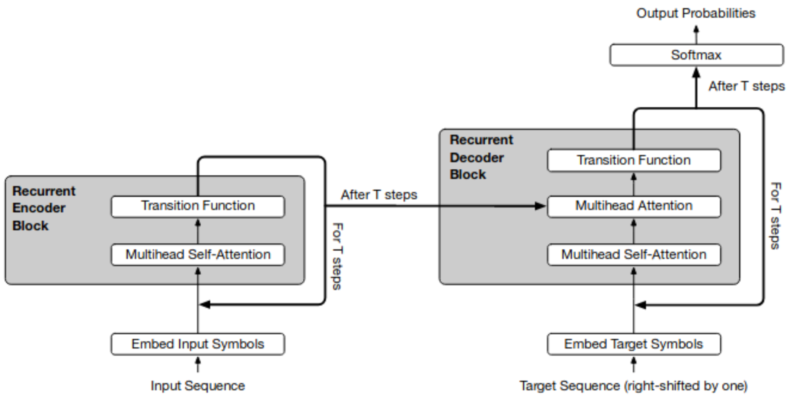


Figure B.1: Universal Transformer. (Dehghani et al., 2019)

These models use weight sharing between layers and operate as a recurrent model. However, instead of the time of a sequence, the computations repeat through the depth (Figure ??). This type of model is called weight-tied input-injected network.

A growing body of literature suggests that recurrent connections improve the learning efficiency and generalization of vision models on classic computer vision challenges Nayebi et al. (2018) and seem to learn visual routines that standard feedforward CNNs do not (Linsley et al. (2018) Kim et al. (2019) Linsley et al. (2020)). Liao and Poggio (2016) discuss relations between Residual Networks (ResNet) He et al. (2015b), Recurrent Neural Networks (RNNs) and the primate visual cortex and observe that a particular type of shallow RNN is precisely equivalent to a very deep ResNet with weight sharing among the layers. A direct implementation of such an RNN, although having orders of magnitude fewer parameters, leads to performance similar to the corresponding ResNet.

Aside from the empirical results, theoretically, Bai et al. (2019a) have proved the universality of Weight-tied Deep Networks which is shown in Theorem B.0.1.

Theorem B.0.1 (Universality of Weight-tied Deep Networks)**.** *Consider a traditional L-layer deep network defined by the relation*

$$\mathbf{z}^{[i+1]} = \sigma^{[i]}\left(W^{[i]}\mathbf{z}^{[i]} + \mathbf{b}^{[i]}\right), \quad i = 0, \ldots, L-1, \quad \mathbf{z}^{[0]} = \mathbf{x}$$

where $\mathbf{z}^{[i]}$ *denotes the hidden features at depth* $i, W^{[i]}, \mathbf{b}^{[i]}$ *are parameters of the network,* $\sigma^{[i]}$ *is the non-linearity at depth* i*, and* x *is the original input. Then the same network can be represented by a weight-tied, input-injected network of equivalent depth*

$$\tilde{\mathbf{z}}^{[i+1]} = \sigma\left(W_z\tilde{\mathbf{z}}^{[i]} + W_x\mathbf{x} + \tilde{\mathbf{b}}\right), \quad i = 0, \ldots, L-1$$

where σ, W_z, W_x *and* $\tilde{\mathrm{b}}$ *are constant over all layers.*

Appendix C

Proofs on chapter 3

C.0.1 Lemma 3.2.1

We prove the lemma for single input case i.e $g_\theta(\mathbf{z}, \mathbf{x}) = \mathbf{z} - \sigma(W\mathbf{z} + U\mathbf{x})$.

Lemma C.0.1. *The partial derivatives of g_θ with respect to $\mathbf{z}, A, B,$ and U are*

$$\begin{aligned}
\frac{\partial g_\theta}{\partial \mathbf{z}} &= [I_n - JW] \,, \\
\frac{\partial g_\theta}{\partial W} &= - J \left[\mathbf{z}^T \otimes I_n\right] \,, \\
\frac{\partial g_\theta}{\partial A} &= - \left[\mathbf{z}^T \otimes J\right] P_A \\
\frac{\partial g_\theta}{\partial B} &= - J \left[\mathbf{z}^T \otimes I_n\right] P_B \\
\frac{\partial g_\theta}{\partial U} &= - J \left[\mathbf{x}^T \otimes I_n\right]
\end{aligned}$$

.

where $P_A = - \left[(I_{nn} + K^{(n)})(I \otimes A^T)\right]$ *and* $P_B = \left[(I_{nn} - K^{(n)})\right]$

Proof. Denote $J \triangleq \operatorname{diag}(\sigma'(W\mathbf{z} + U\mathbf{x}))$.

The differential of g_θ is given by

$$\begin{aligned}
dg_\theta &= d(\mathbf{z} - \sigma(W\mathbf{z} + U\mathbf{x})) \\
&= d\mathbf{z} - Jd(W\mathbf{z} + U\mathbf{x}) \\
&= (I - JW)d\mathbf{z} - J(dW)\mathbf{z} - J(dU)\mathbf{x}
\end{aligned}$$

Taking vectorization on both size using rule $\operatorname{vec}[ABC] = (C^\top \otimes A)\operatorname{vec}[B]$

$$\operatorname{vec}[dg_\theta] =(I - JW)d\mathbf{z} - (\mathbf{z}^T \otimes J)\operatorname{vec}[dW] - (\mathbf{x}^T \otimes J)\operatorname{vec}[dU] \tag{C.1}$$

$$=(I - JW)d\mathbf{z} - J(\mathbf{z}^T \otimes I_n)\operatorname{vec}[dW] - J(\mathbf{x}^T \otimes I_n)\operatorname{vec}[dU] \tag{C.2}$$

Since W is a function of A, B. The differential of W is given by

$$\begin{aligned} dW &= d\left[(1-\mu)I - A^\top A + B - B^\top\right] \\ &= -d(A^TA) + dB - d(B^T) \\ &= -A^TdA - (dA)^TA + dB - (dB)^T \end{aligned}$$

Taking vectorization on both size using rule of commutation matrix. If $K^{(m,n)}$ is a commutation matrix then (1) $K^{(m,n)}\operatorname{vec}[A] = \operatorname{vec}[A^T]$ and (2) $K^{(r,m)}(A \otimes B)K^{(n,q)} = B \otimes A$. In the special case of $m = n$ the commutation matrix is an involution and symmetric matrix. In that case, we simply write $K^{(n)}$ instead of $K^{(n,n)}$. We have

$$\begin{aligned} \operatorname{vec}[dW] &= -(I \otimes A^T)\operatorname{vec}[dA] - (A^T \otimes I)\operatorname{vec}[dA^T] + \operatorname{vec}[dB] - \operatorname{vec}[dB^T] \\ &= -(I \otimes A^T)\operatorname{vec}[dA] - K^{(n)}K^{(n)}(A^T \otimes I)K^{(n)}K^{(n)}\operatorname{vec}[dA^T] + \operatorname{vec}[dB] - K^{(n)}K^{(n)}\operatorname{vec}[dB^T] \\ &= -(I \otimes A^T)\operatorname{vec}[dA] - K^{(n)}K^{(n)}(A^T \otimes I)K^{(n)}\operatorname{vec}[dA] + \operatorname{vec}[dB] - K^{(n)}\operatorname{vec}[dB] \\ &= -\left[(I \otimes A^T) + K^{(n)}(I \otimes A^T)\right]\operatorname{vec}[dA] + (I_{nn} - K^{(n)})\operatorname{vec}[dB] \\ &= -\left[(I_{nn} + K^{(n)})(I \otimes A^T)\right]\operatorname{vec}[dA] + (I_{nn} - K^{(n)})\operatorname{vec}[dB] \\ &= P_A\operatorname{vec}[dA] + P_B\operatorname{vec}[dB] \end{aligned}$$

where $P_A = -\left[(I_{nn} + K^{(n)})(I \otimes A^T)\right]$ and $P_B = (I_{nn} - K^{(n)})$. Plug this into Eq. C.1, we have

$$\operatorname{vec}[dg_\theta] = (I - JW)d\mathbf{z} - J(\mathbf{z}^T \otimes I_n)(P_A\operatorname{vec}[dA] + P_B\operatorname{vec}[dB]) - J(\mathbf{x}^T \otimes I_n)\operatorname{vec}[dU]$$

So

$$\begin{aligned} \frac{\partial g_\theta}{\partial \mathbf{z}} &= [I_n - JW], \\ \frac{\partial g_\theta}{\partial W} &= -J\left[\mathbf{z}^T \otimes I_n\right], \\ \frac{\partial g_\theta}{\partial A} &= -\left[\mathbf{z}^T \otimes J\right]P_A = J\left[\mathbf{z}^T \otimes I_n\right]\left[(I_{nn} + K^{(n)})(I \otimes A^T)\right] \\ \frac{\partial g_\theta}{\partial B} &= -J\left[\mathbf{z}^T \otimes I_n\right]P_B = -J\left[\mathbf{z}^T \otimes I_n\right]\left[(I_{nn} - K^{(n)})\right] \\ \frac{\partial g_\theta}{\partial U} &= -J\left[\mathbf{x}^T \otimes I_n\right] \end{aligned}$$

.

□

C.0.2 Lemma 3.2.2

Lemma C.0.2. *The partial derivatives of Z^* with respect to A, B, and U are*

$$\begin{aligned}
\frac{\partial\,\mathrm{vec}[Z^*]}{\partial\,\mathrm{vec}[W]} &= -\,Q^{-1}D\,[I_m \otimes Z]\,, \\
\frac{\partial\,\mathrm{vec}[Z^*]}{\partial\,\mathrm{vec}[A]} &= Q^{-1}D\,[I_m \otimes Z]\,P_A \\
\frac{\partial\,\mathrm{vec}[Z^*]}{\partial\,\mathrm{vec}[B]} &= -\,Q^{-1}D\,[I_m \otimes Z]\,P_B \\
\frac{\partial\,\mathrm{vec}[Z^*]}{\partial\,\mathrm{vec}[U]} &= -\,Q^{-1}D\,[I_m \otimes X]
\end{aligned}$$

.

where $P_A = (I + K^{(m)})(I_m \otimes A^T), P_B = (I - K^{(m)})$.

C.0.3 Lemma 3.3.3

Lemma C.0.3. $\|Z^*\|_F \leq \frac{1}{m}\|X\|_F\|U\|_2$.

Proof. We will use forward-backward splitting with $Z^0 = 0$ and $\alpha \in \left(0, \frac{2m}{L[I-W]^2}\right)$

$$\begin{aligned}
\|Z^{(l)}\|_F =& \|\sigma(Z^{(l-1)}T_\alpha + \alpha XU)\|_F \\
\leq& \|Z^{(l-1)}T_\alpha + U\|_F \\
\leq& \|Z^{(l-1)}\|_F\|T_\alpha\|_2 + \alpha\|X\|_F\|U\|_2 \\
\leq& \alpha\|X\|_F\|U\|_2 \sum_{i=0}^{l-1} L[T_\alpha]^i \\
\leq& \frac{\alpha}{1 - L[T_\alpha]}\|X\|_F\|U\|_2 \\
\leq& \frac{\alpha}{1 - \sqrt{1 - 2\alpha m + \alpha^2 L[I-W]^2}}\|X\|_F\|U\|_2
\end{aligned}$$

Take $\alpha \to 0^+$ and $l \to \infty$ we have

$$\|Z^*\|_F \leq \frac{1}{m}\|X\|_F\|U\|_2$$

□

C.0.4 Lemma 3.3.2

To prove lemma 3.3.2, we first prove following lemma

Lemma C.0.4. $\|[I - W^\top J]^{-1}\mathbf{u}\| \leq \frac{\|\mathbf{u}\|}{m}$

Proof. Let $\Pi = [J + \mu(I - J)]^{-1}$ with $\mu > 0$. Since $J_{ii} < 1 \forall i \in [n]$, we have $\Pi \in D^+$.

Let
$$\mathbf{v}^* = [I - W^\top J]^{-1}\mathbf{u}. \tag{C.3}$$
This implies $\mathbf{v}^* = W^\top J\mathbf{v}^* + \mathbf{u}$. Since $\Pi \in D^+$, $\mathbf{v}^*$ can have the form $\mathbf{v}^* = \Pi\bar{\mathbf{v}}^*$.

Rewrite C.3,
$$\begin{aligned}
&\Pi\mathbf{v}' = W^\top J\Pi\mathbf{v}' + \mathbf{u}\\
\Leftrightarrow&\Pi\mathbf{v}' + \mu W^\top(I - J)\Pi\mathbf{v}' = W^\top\mathbf{v}' + \mathbf{u}\\
\Leftrightarrow&\left[I + \mu W^\top(I - J)\right]\Pi\mathbf{v}' = W^\top\mathbf{v}' + \mathbf{u}\\
\Leftrightarrow&\left\{\left[I + \mu W^\top(I - J)\right]\Pi - I\right\}\mathbf{v}' + \left(I - W^\top\right)\mathbf{v}' - \mathbf{u} = 0
\end{aligned}$$

Let $G = \left\{\left[I + \mu W^\top(I - J)\right]\Pi - I\right\}$ and $F(\mathbf{v}') = \left(I - W^\top\right)\mathbf{v}' - \mathbf{u}$.

The resolvent operator for G is $R_G = (I+\gamma G)^{-1}$, with $\gamma = 1$ then $R_G = \Pi^{-1}\left[I + \mu W^\top(I - J)\right]^{-1}$.

The update of forward-backward splitting is
$$\bar{\mathbf{v}}^{k+1} = R_G(\bar{\mathbf{v}}^k - \alpha F(\bar{\mathbf{v}}^k)) = \Pi^{-1}\left[I + \mu W^\top(I - J)\right]^{-1}\left\{\left[I - \alpha(I - W^\top)\right]\bar{\mathbf{v}}^k + \alpha\mathbf{u}\right\} \tag{C.4}$$

From the proposition 3, we have $L[\left[I - \alpha(I - W^\top)\right]] \le \sqrt{1 - 2\alpha m + \alpha^2 L[I - W^\top]^2}$. Take the limit $\mu \to 0^+$, we have $\|\Pi\| \le 1$ and $\|R_G\| \le 1$.

Same argument from lemma 3.3.3, we have $\|\bar{\mathbf{v}}^*\| \le \frac{\|\mathbf{u}\|}{m}$. Further,
$$\mathbf{v}^* = \Pi\bar{\mathbf{v}}^* = \left\{\left[I - \alpha(I - W^\top)\right]\bar{\mathbf{v}}^* + \alpha\mathbf{u}\right\}$$
when $\mu \to 0$). So $\|\mathbf{v}^*\| \le \|\bar{\mathbf{v}}^*\| = \frac{\|\mathbf{u}\|}{m}$. □

Lemma C.0.5. $\|(\mathbf{u}^\top \otimes I_N)Q^{-1}\| \le \frac{\|\mathbf{u}\|}{m}$

Proof. We have $(\mathbf{u}^\top \otimes I_N)Q^{-1} = V * I_N$ where $*$ denote row-wise Khatri-Rao product and $V \in \mathbb{R}^{Nn}, V_{i:} = \mathbf{u}^\top[I - J_iW]^{-1}$.

Let $\mathbf{v} \in \mathbb{R}^{Nn}$ and partition $\mathbf{v} = \left[\mathbf{v}_1^\top \dots \mathbf{v}_N^\top\right]^\top$ where $\mathbf{v}_i \in \mathbb{R}^n$. Then $(V * I_N)\mathbf{v} = [V_{1:}\mathbf{v}_1 \dots V_{N:}\mathbf{v}_N]$.

$$\begin{aligned}
\|(V * I_N)\mathbf{v}\|^2 =&\sum_{i=1}^N \langle V_{i:}^\top, \mathbf{v}_i\rangle^2\\
\le&\sum_{i=1}^N \|V_{i:}^\top\|^2\|\mathbf{v}_i\|^2\\
\le&\sum_{i=1}^N \frac{\|\mathbf{u}\|^2}{m^2}\|\mathbf{v}_i\|^2\\
=&\frac{\|\mathbf{u}\|^2}{m^2}\|\mathbf{v}\|^2.
\end{aligned}$$

So $\sup_{\mathbf{v}\in\mathbb{R}^{Nn}} \frac{\|(V*I_N)\mathbf{v}\|}{\|\mathbf{v}\|} \leq \frac{\|\mathbf{u}\|}{m}$ i.e $\|(\mathbf{u}^\top \otimes I_N)Q^{-1}\| \leq \frac{\|\mathbf{u}\|}{m}$. □

C.0.5 Lemma 3.3.4

Lemma C.0.6. *For each $k, s \in [0,T]$, it holds that*

$$\|Z_k - Z_s\|_F \leq \|X\|_F \left(\frac{\|U_k - U_s\|_2}{m} + \frac{\|W_k - W_s\|_2 \|U_s\|_2}{m^2} \right) \tag{C.5}$$

and

$$\|\hat{\mathbf{y}}_k - \hat{\mathbf{y}}_s\| \leq \|X\|_F \left(\frac{\|U_k - U_s\| \|\mathbf{u}_k\|}{m} + \frac{\|W_k - W_s\| \|U_s\| \|\mathbf{u}_k\|}{m^2} + \frac{\|U_s\|}{m} \|\mathbf{u}_k - \mathbf{u}_s\| \right) \tag{C.6}$$

Proof. We unroll iterations of the forward-backward splitting. Let $T_{\alpha,k} = I - \alpha(I - W_k)$.

$$\begin{aligned}
\|Z_k^l - Z_s^l\|_F \leq & \|\sigma(Z_k^{l-1}T_{\alpha,k} + \alpha X U_k) + \sigma(Z_s^{l-1}T_{\alpha,s} + \alpha X U_s)\|_F \\
\leq & \|Z_k^{l-1}T_{\alpha,k} - Z_s^{l-1}T_{\alpha,s} + \alpha X(U_k - U_s)\|_F \\
\leq & \|Z_k^{l-1}T_{\alpha,k} - Z_s^{l-1}T_{\alpha,s}\|_F + \alpha\|X\|_F\|U_k - U_s\|_2 \\
\leq & \|Z_k^{l-1}T_{\alpha,k} - Z_s^{l-1}T_{\alpha,k} + Z_s^{l-1}T_{\alpha,k} - Z_s^{l-1}T_{\alpha,s}\|_F + \alpha\|X\|_F\|U_k - U_s\|_2 \\
\leq & \|Z_k^{l-1} - Z_s^{l-1}\|_F\|T_{\alpha,k}\|_2 + \|Z_s^{l-1}\|_F\|T_{\alpha,k} - T_{\alpha,s}\|_2 + \alpha\|X\|_F\|U_k - U_s\|_2 \\
= & \|Z_k^{l-1} - Z_s^{l-1}\|_F\|T_{\alpha,k}\|_2 + \alpha\|Z_s^{l-1}\|_F\|W_k - W_s\|_2 + \alpha\|X\|_F\|U_k - U_s\|_2 \\
\leq & \|Z_k^{l-1} - Z_s^{l-1}\|_F\|T_{\alpha,k}\|_2 + \frac{\alpha^2\|W_k - W_s\|_2\|X\|_F\|U_s\|_2}{1 - L[T_{\alpha,s}]} + \alpha\|X\|_F\|U_k - U_s\|_2 \\
\leq & \left[\frac{\alpha^2\|W_k - W_s\|_2\|X\|_F\|U_s\|_2}{1 - L[T_{\alpha,s}]} + \alpha\|X\|_F\|U_k - U_s\|_2\right] \sum_{i=0}^{l-1} \|T_{\alpha,k}\|_2^i
\end{aligned}$$

The above inequality holds for all l. Taking the limit as $l \to \infty$, we have

$$\begin{aligned}
& \|Z_k - Z_s\|_F = \lim_{l\to\infty} \|Z_k^l - Z_s^l\|_F \\
\leq & \left[\frac{\alpha^2\|W_k - W_s\|_2\|X\|_F\|U_s\|_2}{1 - L[T_{\alpha,s}]} + \alpha\|X\|_F\|U_k - U_s\|_2\right] \sum_{i=0}^{\infty} \|T_{\alpha,k}\|_2^i \\
\leq & \frac{\alpha^2\|W_k - W_s\|_2\|X\|_F\|U_s\|_2}{(1 - L[T_{\alpha,s}])(1 - L[T_{\alpha,k}])} + \frac{\alpha\|X\|_F\|U_k - U_s\|_2}{1 - L[T_{\alpha,k}]}
\end{aligned}$$

Finally, take $\alpha \to 0$, we have that

$$\|Z_k - Z_s\|_F \leq \|X\|_F \left(\frac{\|U_k - U_s\|_2}{m} + \frac{\|W_k - W_s\|_2 \|U_s\|_2}{m^2} \right)$$

We have proven the first part. For the second part

$$\begin{aligned}
\|\hat{\mathbf{y}}_k - \hat{\mathbf{y}}_s\|_2 =& \|Z_k\mathbf{u}_k - Z_s\mathbf{u}_s\|_2 \\
=& \|Z_k\mathbf{u}_k - Z_s\mathbf{u}_k + Z_s\mathbf{u}_k - Z_s\mathbf{u}_s\|_2 \\
\leq& \|(Z_k - Z_s)\|_2\|\mathbf{u}_k\| + \|Z_s\|_2(\mathbf{u}_k - \mathbf{u}_s)\|_2 \\
\leq& \|(Z_k - Z_s)\|_F\|\mathbf{u}_k\| + \|Z_s\|_F(\mathbf{u}_k - \mathbf{u}_s)\|_2 \\
\leq& \|X\|_F\left(\frac{\|U_k - U_s\|_2}{m} + \frac{\|W_k - W_s\|_2\|U_s\|_2}{m^2}\right)\|\mathbf{u}_k\| + \|X\|_F\frac{\|U_s\|}{m}\|\mathbf{u}_k - \mathbf{u}_s\|_2 \\
\leq& \|X\|_F\left(\frac{\|U_k - U_s\|\|\mathbf{u}_k\|}{m} + \frac{\|W_k - W_s\|\|U_s\|\|\mathbf{u}_k\|}{m^2} + \frac{\|U_s\|}{m}\|\mathbf{u}_k - \mathbf{u}_s\|\right)
\end{aligned}$$

□