1-Lipschitz Deep Equilibrium Models

Abhay Singh
Cornell University
as2626@cornell.edu

Aaron Lou
Cornell University
al968@cornell.edu

Abstract

Deep Equilibrium Models (DEQs) are a powerful class of neural networks which represent a fixed point solution to an iterative solver. To train our model using gradient descent, we can use implicit differentiation on the fixed point formula. However, in practice, the neural networks need to be constrained such that the fixed point solution always exists. In this paper, we present a method for enforcing the uniqueness of the fixed point solution using 1-Lipschitzness of the neural network. To do this, we parameterize our weight matrices with Orthogonal matrices using Lie Theory and use 1-Lipschitz activation functions like groupsort. We apply our method to preexisting DEQs for image and graph data and find that our constraints improve performance and stability.

1 Introduction

A deep neural network often is described in terms of a composition of L layers. Each of the L layers repeatedly transform the input from the last layer into a corresponding output; we refer to the 'depth' of such a network as L. To perform backpropagation in order to update the weights of this model, the backward pass stores the intermediate outputs of the L layers. L is a hyperparameter chosen by model designers. In the simplest case such a feed-forward model can be written as follows

$$\mathbf{z}^{[i+1]} = f_\theta^{[i]}(\mathbf{z}^{[i]}; \mathbf{x}) \qquad \text{for } i = 0, 1, 2, \ldots, L-1. \tag{1}$$

Many of deep learning successes' has been attributed to overparameterized deep networks, where L is large, so that we have many different $f_\theta^{[i]}$ with their own parameters. However, if we instead to share weights along all our layers such that $f_\theta^{[i]} = f_\theta$ for $i = 0, 1, 2, \ldots, L-1$, it has been studied that we can achieve similar performance. If we further consider the limit of this model as $L \to \infty$, we obtain the Deep Equilibrium Model (DEQ) [3] method to directly compute a fixed point $\mathbf{z}$ of the non-linear transformation

$$\mathbf{z}^* = f_\theta(\mathbf{z}^*; \mathbf{x}). \tag{2}$$

However this method, without any other checks, suffers from issues of convergence leading to suboptimal performance. In our work, we constrain f_θ to be 1-Lipschitz, i.e. $\|f_\theta(\mathbf{z}_1; \mathbf{x}) - f_\theta(\mathbf{z}_2; \mathbf{x})\| \leq \|\mathbf{z}_1 - \mathbf{z}_2\|$. Then by Banach's fixed point theorem, we have that $\mathbf{z}^*$ exists and is unique. Now we have an 'infinitely' deep model with good training guarantees.

In order to constrain f_θ to be 1-Lipschitz, we utilize orthogonal weight matrices from [5] and orthogonal convolutions from [9]. Further we use the 1-Lipschitz activation function GroupSort proposed in [1]. Finally we remove normalization layers such as BatchNorm and use convex combinations of inputs. Our general method shows improved training dynamics and converges reliably to a fixed-point. When dropping in our method on graph and node classification tasks we outperform previous existing methods, [6], in performance.

Preprint. Under review.

2 Related Work

2.1 Deep Equilibirum Models

Originally introduced in [3], DEQs have experienced a surge in popularity both from a theoretical and empirical lens. Theoretically, many papers have analyzed sufficient conditions to converge to the fixed point solution. For example various papers such as [**?**] have sought to explicitly enforce several constraints to satisfy the DEQ conditions, while others have developed faster methods to solve the fixed point problem. Alternatively, the core idea of the DEQ has been reapplied to a variety of other domains. For example, [4] applied the DEQ to large and hierachical data, and [6] applied the DEQ to graph neural networks.

2.2 1-Lipschitz Neural Networks

Originally developed for applications to Wasserstein GAN [2], 1-Lipschitz neural networks have found a variety of other uses in fields such as theoretically principled defences against adversarial attacks [1]. The core idea is to 1) parameterize the weight matrices to be 1-Lipschitz and 2) use 1-Lipschitz activation functions. The original construction applied Björck Orthonormalization to each weight (before forwarding through the neural network) and used the groupsort activation function to maintain 1-Lipschitzness.

2.3 Cayley Convolution

The Cayley convolution [9] work orthogonalizes a skew-symmetric convolution using the Cayley transform in the Fourier domain. This allows the inverse convolution, which is used in computing the Cayley transform, to be computed efficiently. The authors also discuss norm-preservation, albeit in the adversarial robustness case rather than for a DEQ model.

2.4 Optimization on Lie Groups

When optimizing over orthogonal matrices, it is useful to consider our viable space as a Lie Group. This theoretical framework allows us to apply existing optimization methods such as Riemannian optimization [8], which greatly simplifies the construction. For our purposes, we require a fast and efficient methodology which works well with an automatic differentiation framework [7]. To this end, we use the methodology outlined in Cheap Orthogonal Constraints [5] which parameterizes the Lie Group of orthogonal matrices through the local tangent spaces (specifically the transported Lie Algebra). This framework is easily integrated with regular autograd as all optimization takes places on a Euclidean space (as opposed to the nonlinear Riemannian manifolds), and in particular allows us to seamlessly upgrade the previous approaches.

3 Methodology

3.1 Background

An matrix $Q \in \mathbb{R}^{m \times n}$ where $m = n$ is said to be orthogonal if $Q^T Q = QQ^T = I$ the identity. However in a neural network, it is more often the case that $m \neq n$. In this case, if Q satisfies the above, we have semi-orthogonality. In the case where $m\ gen$ we have norm-preservation, i.e. for any $x \in \mathbb{R}^n$, $\|Qx\|_2 = \|x\|_2$. On the other hand if $m < n$ we have contractiveness, i.e. for any $x \in \mathbb{R}^n$, $\|Qx\|_2 \leq \|x\|_2$.

Some other useful properties we have are that for two functions f, g their composition $\|f \circ g\|_{\text{Lip}} \leq \|f\|_{\text{Lip}} \|g\|_{\text{Lip}}$ and further $\|f + g\|_{\text{Lip}} \leq \|f\|_{\text{Lip}} + \|g\|_{\text{Lip}}$. These are useful for reasoning about additional applications of layers as well as residual connections, where multiple inputs are used in an output of a network.

3.2 Construction

To parameterize a 1-Lipschitz neural network, we let the i-th layer be defined as

$$L^i(x) = \sigma(W_{or}^i x + b^i)$$

where W_{or} is a orthogonal matrix and σ is the 1-Lipschitz function groupsort [1]. Similarly, we also will not apply an activation function to the last layer L^n.

To construct the W_{or} we use the orthogonal matrix construction in [5]. This parameterizes the Lie Group of orthogonal matrices with the exponential map, so $W_{or} = \exp(w_{or})$ where $w_{or} \in \mathfrak{o}(n)$. Because the Lie Group is connected and compact, the exponential map is surjective and so we are able to represent all orthogonal matrices. Furthermore, since $\mathfrak{o}(n)$ is Euclidean, we are able to optimize over all Euclidean parameters.

We then scale the resulting output by some $\kappa \in [0, 1)$. Our final DEQ is then is thus given by

$$x^* = \kappa f_{nn}(x^*)$$

and this can be solved using the standard methods (such as Newton's method).

3.3 Theoretical Justification

We first show that κ-Lipschitz neural network are guaranteed to converge to a unique fixed point iterate when $\kappa < 1$.

Prop 3.1. *If $f : \mathbb{R}^d \to \mathbb{R}^d$ is a κ-Lipschitz function where $0 \leq \kappa < 1$, then there exists a unique point x^* s.t. $f(x^*) = x^*$.*

Proof. Note that $|f(x) - f(y)| \leq \kappa|x - y|$ so f is a contraction. By Banach's Fixed Point Theorem, there must exists a unique fixed point x^*. Furthermore, we note that if $x_{n+1} = f(x_n)$ then $x_n \xrightarrow{n\to\infty} x^*$. □

We also note that our restricted neural networks (defined above) are 1-Lipschitz

Prop 3.2. *If W is an orthogonal matrix, then $f(x) = Wx$ is a 1-Lipschitz function.*

Proof. This follows from the adjointness condition of the transpose in inner products. In particular, we see that $\|Wx\|_2 = \sqrt{\langle Wx, Wx\rangle} = \sqrt{\langle x, W^\top Wx\rangle} = \sqrt{\langle x, x\rangle} = \|x\|_2$. □

Prop 3.3. *Since all components are 1-Lipschitz, this implies that our neural network $f_{nn}(x)$ constructed above is a 1-Lipschitz function.*

4 Experiments

We use open-source implementations of [6] and swap in orthogonal weights and norm-preserving activation functions. This consistently improves their methods performance as shown in Table 1. In order to ensure contractiveness, we also multiply the final output $\mathbf{z}^*$ by a value $\kappa \in [0, 1)$. This helps during training since an exactly 1-Lipschitz network would sometimes not converge in beginning epochs due to floating point errors and other non-determinism; therefore we actually have a κ-Lipschitz network.

Table 1: Performance on Graph Classification Datasets

	Implicit GNN	1-Lipschitz Implicit GNN
MUTAG	82.9 ± 9.5	89.9 ± 8.2
Proteins	78.4 ± 0.9	79.1 ± 1.1
PTC	66.7 ± 2.4	68.2 ± 2.3
COX2	84.7 ± 4.8	85.7 ± 4.1
NCI1	74.9 ± 2.0	76.6 ± 1.6

Moreover we see improved training stability. We are able to achieve our peak accuracy in around 250 epochs of training, whereas their method takes around 500 epochs.

Further we try computer vision tasks, performing image classification on the CIFAR-10 dataset. We experiment with different tolerance values ϵ for convergence when training to a fixed point, namely ϵ=1e-2, 1e-3, 1e-4. We also vary our value of κ as mentioned before. We train 50 epochs of a single DEQ layer ResNet.

Table 2: Performance on CIFAR-10 after 50 Training Epochs

	Training Error	Test Error
κ=0.95, ϵ=1e-2	0.2018	0.2491
κ=0.90, ϵ=1e-3	0.2011	0.2475
κ=0.80, ϵ=1e-4	0.2045	0.2512
κ=0.95, ϵ=1e-3	0.2004	0.2475
κ=0.98, ϵ=1e-2	0.1935	0.2432
κ=0.95, ϵ=1e-3	0.1918	0.2445

Our results show in Table 2 us that in order to both lowering tolerance value for convergence, ϵ, and also making our network that much more contractive, i.e. use a lower value of κ, maintain similar performance. This intuitively makes sense since this would require a similar number of fixed point iterations. Moreover, we see that enforcing both a high value of κ and a low value of ϵ allows us to improve both train and test performance.

5 Conclusion

We have seen the usefulness of such 1-Lipschitz DEQ models. We have been able to improve performance on existing networks such as that in [6] by plugging in our method, and further we have studied how the training dynamics change as our fixed point-dependent parameters ϵ and κ change. For future work, we will extend the study to deep language models such as Transformers.

References

[1] Cem Anil, James Lucas, and Roger Grosse. Sorting out lipschitz function approximation, 2019.

[2] Martín Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *ArXiv*, abs/1701.07875, 2017.

[3] Shaojie Bai, J. Z. Kolter, and V. Koltun. Deep equilibrium models. In *NeurIPS*, 2019.

[4] Shaojie Bai, V. Koltun, and J. Z. Kolter. Multiscale deep equilibrium models. *ArXiv*, abs/2006.08656, 2020.

[5] Mario Lezcano Casado and David Martínez-Rubio. Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. *ArXiv*, abs/1901.08428, 2019.

[6] Fangda Gu, Heng Chang, Wenwu Zhu, S. Sojoudi, and L. Ghaoui. Implicit graph neural networks. *ArXiv*, abs/2009.06211, 2020.

[7] Adam Paszke, S. Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zach DeVito, Zeming Lin, Alban Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[8] W. Ring and Benedikt Wirth. Optimization methods on riemannian manifolds and their application to shape space. *SIAM J. Optim.*, 22:596–627, 2012.

[9] Asher Trockman and J. Z. Kolter. Orthogonalizing convolutional layers with the cayley transform. *ArXiv*, abs/2104.07167, 2021.