Lipschitz Multiscale Deep Equilibrium Models: A Theoretically Guaranteed and Accelerated Approach

Naoki Sato
Meiji University

Hideaki Iiduka
Meiji University

Abstract

Deep equilibrium models (DEQs) achieve infinitely deep network representations without stacking layers by exploring fixed points of layer transformations in neural networks. Such models constitute an innovative approach that achieves performance comparable to state-of-the-art methods in many large-scale numerical experiments, despite requiring significantly less memory. However, DEQs face the challenge of requiring vastly more computational time for training and inference than conventional methods, as they repeatedly perform fixed-point iterations with no convergence guarantee upon each input. Therefore, this study explored an approach to improve fixed-point convergence and consequently reduce computational time by restructuring the model architecture to guarantee fixed-point convergence. Our proposed approach for image classification, Lipschitz multiscale DEQ, has theoretically guaranteed fixed-point convergence for both forward and backward passes by hyperparameter adjustment, achieving up to a 4.75× speed-up in numerical experiments on CIFAR-10 at the cost of a minor drop in accuracy.

1 INTRODUCTION

The past decade has seen remarkable progress in explicit deep learning, largely driven by the development of increasingly deep neural networks. Architectures like ResNet [He et al., 2016] and Transformers [Vaswani et al., 2017] have demonstrated that performance on a wide range of tasks consistently improves with model depth. This "deeper is better" paradigm has led to models with tens or even hundreds of explicit layers. However, this pursuit of depth comes at a significant cost: (1) substantial memory requirements for storing activations of all intermediate layers for backpropagation, and (2) diminishing returns in performance, where simply stacking more layers can lead to vanishing gradients and optimization difficulties. These challenges have motivated a search for alternative ways to conceptualize and leverage depth in neural networks.

Deep equilibrium models (DEQs) [Bai et al., 2019a,Bai et al., 2020] offer a radical departure from the explicit, layer-stacking approach. Instead of defining a finite sequence of transformations, a DEQ layer conceptualizes its forward pass as the solution to a fixed-point equation. By iteratively applying a single, weight-tied nonlinear transformation until it reaches an equilibrium point, a DEQ effectively creates a neural network of infinite depth. This implicit formulation carries a profound advantage: a drastically reduced memory footprint for training. Unlike explicit N-layer models that require $\mathcal{O}(N)$ memory to store intermediate activations for backpropagation, a DEQ's memory requirement is $\mathcal{O}(1)$ and remains constant, effectively decoupling model depth from memory cost. This allows for theoretically infinite-depth representations without the memory constraints that plague traditional deep networks. Impressively, both the DEQ-Transformer [Bai et al., 2019a] for text tasks and multiscale DEQs (MDEQs) [Bai et al., 2020] for computer vision tasks achieve this excellent memory efficiency while delivering performance comparable to state-of-the-art models.

Despite the promise of DEQs, their practical adoption is severely hindered by a singular, critical issue: prohibitive training and inference times. Empirical results show that these models can be 2.5 to 3 times slower than their explicit counterparts [Bai et al., 2019a, Bai et al., 2020]. We argue that this computational burden is a direct symptom of a deeper theoretical problem:

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. Copyright 2026 by the author(s).

the lack of a guaranteed unique solution to the fixed-point problem they aim to solve. For each new input, a DEQ must solve the following fixed-point problem using some fixed-point solver.

$$\text{Find } \boldsymbol{z} \in \mathrm{Fix}(f_{\boldsymbol{\theta}}) := \left\{ \boldsymbol{z} \in \mathbb{R}^d \colon f_{\boldsymbol{\theta}}(\boldsymbol{z}; \boldsymbol{x}) = \boldsymbol{z} \right\}, \quad (1)$$

where $\boldsymbol{z}$ is hidden states, $\boldsymbol{x}$ is input, $\boldsymbol{\theta}$ is weight parameters, and $f_{\boldsymbol{\theta}} \colon \mathbb{R}^d \to \mathbb{R}^d$ is the single, repeated transformation. From the Banach fixed-point theorem [Banach, 1922, Kreyszig, 1978], a unique solution to this problem is guaranteed to exist only if the mapping $f_{\boldsymbol{\theta}}$ is a contraction mapping. Here, a mapping $f \colon \mathbb{R}^d \to \mathbb{R}^d$ is said to be a contraction mapping if there exists $r \in (0, 1)$ such that, for all $\boldsymbol{z}_1, \boldsymbol{z}_2 \in \mathbb{R}^d$, $\|f(\boldsymbol{z}_1) - f(\boldsymbol{z}_2)\| \leq r\|\boldsymbol{z}_1 - \boldsymbol{z}_2\|$. In other words, being a contraction mapping is equivalent to being an L-Lipschitz mapping for $L < 1$. It is known that if a mapping f is a contraction mapping, then the sequence generated by the Banach fixed-point approximation method, $\boldsymbol{z}_{t+1} := f(\boldsymbol{z}_t)$, converges to a unique fixed point of f. However, $f_{\boldsymbol{\theta}}$ does not generally satisfy this contractility, because $f_{\boldsymbol{\theta}}$ is a complex combination of convolutions, regularization, nonlinear activations, etc. As a result, fixed-point solvers are forced to perform iterative processes with no convergence guarantee, resulting in unpredictable and often explosive computational costs for both training and inference.

In the present study, our aim was to redesign the model architecture (i.e., $f_{\boldsymbol{\theta}}$) so that there exists a unique solution to the fixed-point problem (1), thereby providing theoretical guarantees for fixed-point convergence, accelerating convergence, and reducing the enormous computational time required for DEQ training and inference. Our approach is rooted in directly controlling the Lipschitz constant of the fixed-point mapping. While the principles of our approach could be generalized, the model structures of DEQ variants like the DEQ-Transformer and MDEQ are fundamentally different. Therefore, we focus here exclusively on MDEQ for computer vision tasks to rigorously validate the effectiveness of our techniques. Our contribution can be summarized as follows:

1. We propose a new variant of MDEQ, Lipschitz MDEQ, which incorporates several modifications to the MDEQ architecture. In contrast to MDEQ, the Lipschitz constant of the fixed-point mapping appearing in Lipschitz MDEQ is composed of several hyperparameters that can be determined by the user. **(Section 3.2)**
2. We demonstrate a combination of hyperparameters that ensures the fixed-point mappings appearing in both the forward pass and backward pass of Lipschitz MDEQ is a contraction mapping. Consequently, Lipschitz MDEQ is a member of the DEQ family that theoretically guarantees fixed-point convergence. **(Section 3.3)**
3. Our numerical experiments demonstrate that Lipschitz MDEQ significantly improves fixed-point convergence, reduces MDEQ computation time by up to 4.75-fold with theoretical guarantees, and enables trade-offs between accuracy and speed by adjusting the Lipschitz constant. **(Section 4)**

2 BACKGROUND

For related work on other theory, applications, adversarial robustness, the implicit layers family, and Lipschitz networks, see Appendix A.

Weight-tied Models DEQs originate from the breakthroughs of models such as TrellisNet [Bai et al., 2019b] and the Universal Transformer [Dehghani et al., 2019], which have shared weights across all layers. Although these models achieve constant memory in the sense that the number of parameters does not increase with the number of layers, they still have the drawback of the computational cost being proportional to the number of layers, due to backpropagation. Bai et al. observed experimentally that the output of the intermediate layer of these models, i.e., the hidden state $\boldsymbol{z}$, converges to a fixed point of the transformation $f_{\boldsymbol{\theta}}$ as the number of layers increases [Bai et al., 2019a]. They proposed the DEQ approach, shifting to directly finding the fixed point $\boldsymbol{z}^\star$ of $f_{\boldsymbol{\theta}}$ rather than adding layers to obtain it. This approach was incorporated into the transformer and demonstrated extremely powerful performance in large-scale sequence modeling tasks [Bai et al., 2019a]. Soon after, MDEQ, incorporating a multiscale mechanism to make it applicable to computer vision tasks, was proposed and shown to demonstrate extremely good performance in both image classification and semantic segmentation tasks [Bai et al., 2020].

Deep Equilibrium Models A DEQ receives an input and solves the fixed-point problem (1) using some fixed-point solver in the forward pass. The number of iterations by the solver corresponds to the number of layers in the DEQ. Once the fixed point $\boldsymbol{z}^\star$ is found, it is used to define an empirical loss function $\ell_{\boldsymbol{z}^\star}(\boldsymbol{\theta})$, and the loss minimization procedure proceeds using a standard optimizer such as stochastic gradient descent or Adam. The required gradient can be obtained in the following form by applying the chain rule and implicit function theorem [Krantz and Parks, 2012]:

$$\frac{\partial \ell_{\boldsymbol{z}^\star}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \underbrace{\frac{\partial \ell_{\boldsymbol{z}^\star}(\boldsymbol{\theta})}{\partial \boldsymbol{z}^\star}(I - J_{f_{\boldsymbol{\theta}}}(\boldsymbol{z}^\star))^{-1}}_{=:A} \frac{\partial f_{\boldsymbol{\theta}}(\boldsymbol{z}^\star; \boldsymbol{x})}{\partial \boldsymbol{\theta}}, \quad (2)$$

where $J_{f_{\boldsymbol{\theta}}}$ is the Jacobian matrix of $f_{\boldsymbol{\theta}}$ at $\boldsymbol{z}^\star$. To avoid the expensive computation of the inverse of the Jacobian matrix, the term A is calculated in the backward pass by solving the following equation:

$$\mathbf{x}\,(I - J_{f_{\boldsymbol{\theta}}}(\boldsymbol{z}^\star)) - \frac{\partial \ell_{\boldsymbol{z}^\star}(\boldsymbol{\theta})}{\partial \boldsymbol{z}^\star} = \mathbf{0}. \tag{3}$$

The first term can be efficiently computed for any $\mathbf{x}$ via autograd packages such as PyTorch [Paszke et al., 2019]. The linear equation (3) is equivalent to the problem of finding a fixed point that satisfies

$$\mathbf{x} = T(\mathbf{x}) \text{ under } T(\mathbf{x}) := \mathbf{x} J_{f_{\boldsymbol{\theta}}}(\boldsymbol{z}^\star) + \frac{\partial \ell_{\boldsymbol{z}^\star}(\boldsymbol{\theta})}{\partial \boldsymbol{z}^\star}. \tag{4}$$

In other words, DEQs solve two different fixed-point problems in the backward and forward passes. In both fixed-point problems, the initial research used the Broyden's method [Broyden, 1965] as the solver, but recent research has focused on the Anderson acceleration [Anderson, 1965], which is provably equivalent to a multi-secant quasi-Newton method [Fang and Saad, 2009], has become mainstream. In addition, some DEQ variants [Pabbaraju et al., 2021, Gabor et al., 2024] use the Peaceman-Rachford method [Peaceman and Rachford, 1955] or the Banach fixed-point approximation method [Banach, 1922] under different formulations. Recently, a RevDEQs that enables the use of backpropagation in the backward pass were proposed that use a reversible fixed-point solver [McCallum et al., 2025].

Theory Generally, for the fixed-point problems solved by DEQs, the existence and uniqueness of fixed points are not guaranteed. Several prior studies have proposed variants of DEQs to ensure these properties. Winston et al. introduced Monotone Operator Equilibrium Networks (monDEQ), ensuring the existence of a fixed point by designing the layer as a monotone operator [Winston and Kolter, 2020]. Revay et al. achieved uniqueness by enforcing contraction under a learnable non-Euclidean norm, which relaxes constraints on the linear layers [Revay et al., 2020]. More recently, Sittoni et al. proposed Subhomogeneous DEQ (SubDEQ), guaranteeing the existence and uniqueness of a bounded solution by imposing subhomogeneity on the mapping [Sittoni and Tudisco, 2024]. Concurrently, Gabor et al. introduced Positive Concave DEQ (pcDEQ), which ensures a unique fixed point by decomposing the function into contractive and non-expansive parts [Gabor et al., 2024].

Acceleration Techniques DEQs are also known for their extremely low training and inference speeds. To address this issue, several prior studies have successfully reduced the computational cost of the backward pass (2) by utilizing approximate gradients [Geng et al., 2021, Ramzi et al., 2022, Fung et al., 2022, Nguyen and Mauch, 2022]. In particular, Fung et al.'s Jacobian-free backpropagation (JFB) achieves acceleration by replacing $(I - J_{f_{\boldsymbol{\theta}}}(\boldsymbol{z}^\star))^{-1}$ with the identity matrix, whereas Geng et al. achieved acceleration by using the phantom gradient of a surrogate loss $\frac{1}{2}\|f_{\boldsymbol{\theta}}(\boldsymbol{z}^\star) - \boldsymbol{z}^\star\|^2$, which is easier to compute [Geng et al., 2021]. Additionally, there are techniques that utilize differences in convergence rates between dimensions within the hidden dimensions of the DEQ to omit computations [Wang et al., 2024a], as well as methods that introduce learnable neural fixed-point solvers to streamline fixed-point iterations [Bai et al., 2022b]. While slightly different in nature, Gurumurthy et al.'s proposed method significantly reduces computation time by simultaneously solving fixed-point iterations and input optimization for specific tasks [Gurumurthy et al., 2021]. From the perspective of accelerating DEQs, the prior study closest to our approach is that proposing Jacobian regularization [Bai et al., 2021]. While Jacobian regularization successfully stabilizes training by the employment of Jacobian norm $\|J_{f_{\boldsymbol{\theta}}}(\boldsymbol{z}^\star)\|_{\mathrm{F}}$ as a penalty term for the backward pass, our approach achieves acceleration grounded in the theory of both the forward and backward passes by constraining the supremum of $\|J_{f_{\boldsymbol{\theta}}}(\boldsymbol{z})\|_2$ to be less than 1. Furthermore, although the Jacobian regularization approach does not have theoretical convergence guarantees, it has the practical advantage of incurring minimal performance degradation. Note that Jacobian regularization stabilizes fixed-point convergence to achieve high accuracy with fewer iterations, but it does not guarantee or accelerate fixed-point convergence.

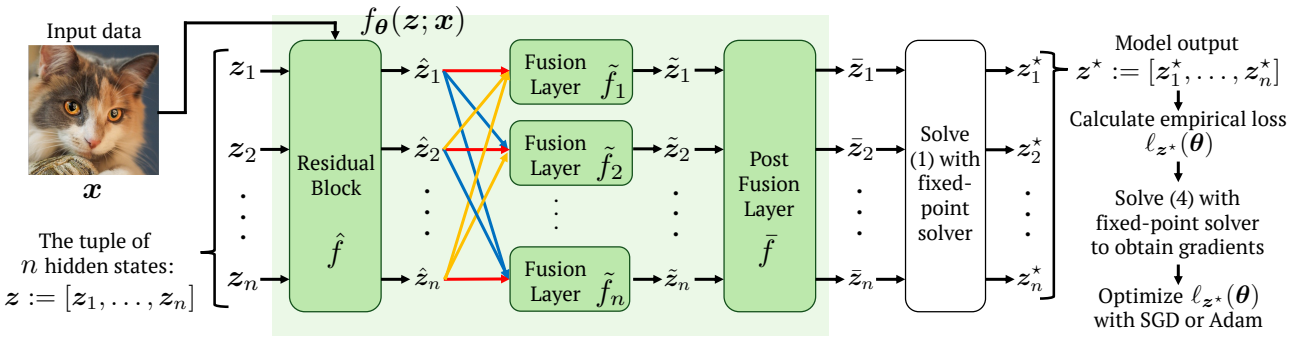
3 MAIN RESULTS

3.1 Preliminaries

Notation Let $\mathbb{N}$ be the set of non-negative integers. For $n \in \mathbb{N}$, define $[n] := \{1, 2, \ldots, n\}$. Let $\mathbb{R}^d$ be a d-dimensional Euclidean space with inner product $\langle \cdot, \cdot \rangle$, which induces the norm $\|\cdot\|$. Let $f \circ g$ be the composite mapping of f and g, i.e., $(f \circ g)(\cdot) := f(g(\cdot))$. Here, f^k will denote applying mapping f k times, with f^0 denoting the identity mapping. We use $\odot$ to denote the element-wise product, also known as the Hadamard product.

Definition 3.1 (L-Lipschitz mapping)**.** *Given a mapping $f\colon \mathbb{R}^d \to \mathbb{R}^d$, if $\|f(\boldsymbol{x}) - f(\boldsymbol{y})\| \leq L_f \|\boldsymbol{x} - \boldsymbol{y}\|$ holds for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, then f is said to be an L_f-Lipschitz mapping, and L_f is called the Lipschitz constant of f.*

As in Definition 3.1, we use L_f to represent the Lipschitz constant of a mapping f. For specific operators, we use simplified versions such as L_{Conv} and L_{ReLU}.



MDEQ Architecture [Bai et al., 2020]

$$\begin{cases} \hat{\boldsymbol{z}}_i = \hat{f}_i(\boldsymbol{z}_i) := \mathrm{GN}(\mathrm{ReLU}(\boldsymbol{z}_i + \mathrm{GN}(\mathrm{Drop}(\mathrm{Conv}(\mathrm{ReLU}(\mathrm{GN}(\mathrm{Conv}(\boldsymbol{z}_i))))))))) & \text{(M1)} \\ \tilde{\boldsymbol{z}}_i = \tilde{f}_i(\hat{\boldsymbol{z}}_i) := \hat{\boldsymbol{z}}_i + \sum_{j=1}^{i-1} \mathrm{D}_{i,j}(\hat{\boldsymbol{z}}_i) + \sum_{j=i+1}^{n} \mathrm{U}_{i,j}(\hat{\boldsymbol{z}}_i) & \text{(M2)} \\ \bar{\boldsymbol{z}}_i = \bar{f}_i(\tilde{\boldsymbol{z}}_i) := \mathrm{GN}(\mathrm{Conv}(\mathrm{ReLU}(\tilde{\boldsymbol{z}}_i))) & \text{(M3)} \end{cases}$$

Lipschitz-MDEQ Architecture (ours)

$$\begin{cases} \hat{\boldsymbol{z}}_i = \hat{f}_i(\boldsymbol{z}_i) := \mathrm{MGN}(\mathrm{SReLU}((1-\alpha_1)\boldsymbol{z}_i + \alpha_1 \mathrm{MGN}(\mathrm{Drop}(\mathrm{Conv}^\star(\mathrm{SReLU}(\mathrm{MGN}(\mathrm{Conv}^\star(\boldsymbol{z}_i))))))))) & \text{(L1)} \\ \tilde{\boldsymbol{z}}_i = \tilde{f}_i(\hat{\boldsymbol{z}}_i) := (1-\alpha_2)\hat{\boldsymbol{z}}_i + \alpha_2 \left(\sum_{j=1}^{i-1} w_{ij} \mathrm{D}_{i,j}(\hat{\boldsymbol{z}}_i) + \sum_{j=i+1}^{n} w_{ij} \mathrm{U}_{i,j}(\hat{\boldsymbol{z}}_i) \right) & \text{(L2)} \\ \bar{\boldsymbol{z}}_i = \bar{f}_i(\tilde{\boldsymbol{z}}_i) := \mathrm{MGN}(\mathrm{Conv}^\star(\mathrm{SReLU}(\tilde{\boldsymbol{z}}_i))) & \text{(L3)} \end{cases}$$

Figure 1: Architecture of MDEQ and Lipschitz MDEQ. Lipschitz MDEQ inherits the general structure of MDEQ, with each operation modified to prevent the Lipschitz constant from becoming too large. Note that the layout and notation of the figure are taken from the prior work [Bai et al., 2020, Figure 1] with only minor modifications made for clarity in explaining our results. The sample image is from the AFHQ dataset [Choi et al., 2020].

The following lemma is frequently used to derive the Lipschitz constant for the forward pass of MDEQ, and is here used for the proposed Lipschitz MDEQ.

Lemma 3.1. *Let the mappings $f\colon \mathbb{R}^d \to \mathbb{R}^d$ and $g\colon \mathbb{R}^d \to \mathbb{R}^d$ be L_f-Lipschitz and L_g-Lipschitz mappings, respectively. Then, the mapping $f+g$ is an (L_f+L_g)-Lipschitz mapping. Also, the mapping $f \circ g$ is an $L_f L_g$-Lipschitz mapping.*

First, we clarify how the fixed-point map $f_{\boldsymbol{\theta}}$ appearing in MDEQ is defined. In MDEQ, hidden states are defined for each resolution, processed in parallel, and influence each other. Let the number of resolutions be $n \in \mathbb{N}$, and let $\boldsymbol{z}_1$ and $\boldsymbol{z}_n$ denote the hidden states at the highest and lowest resolutions, respectively. The input image is injected only into $\boldsymbol{z}_1$, and the other hidden states receive input only indirectly. However, note that since we are considering the Lipschitz constant of $f_{\boldsymbol{\theta}}$ with respect to the hidden state $\boldsymbol{z} := [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n]$, the processing of the input can be ignored. $f_{\boldsymbol{\theta}}$ consists of three blocks: a residual block $\hat{f}\colon \boldsymbol{z} \mapsto \hat{\boldsymbol{z}}$ (Eq. (M1)), a fusion layer $\tilde{f}\colon \hat{\boldsymbol{z}} \mapsto \tilde{\boldsymbol{z}}$ (Eq. (M2)), and a post-fusion layer $\bar{f}\colon \tilde{\boldsymbol{z}} \mapsto \bar{\boldsymbol{z}}$ (Eq. (M3)) (see Figure 1), noting that $f_{\boldsymbol{\theta}} = \bar{f} \circ \tilde{f} \circ \hat{f}$. The operations constituting these will be detailed in Section 3.2 along with our modifications.

3.2 Lipschitz MDEQ Architecture

We will detail each operation appearing in MDEQ and derive its Lipschitz constant. Then we will modify each operation so that the Lipschitz constant of the mapping f_θ is less than 1, thereby constructing the Lipschitz MDEQ.

Normalization MDEQ employs Group Normalization (GN) [Wu and He, 2018], which normalizes features within groups of channels. For an input group of features $\boldsymbol{z}$, the standard GN operation is defined as

$$\mathrm{GN}(\boldsymbol{z})_i = \gamma \frac{z_i - \mu_{\boldsymbol{z}}}{\sqrt{\sigma_{\boldsymbol{z}}^2 + \epsilon}} + \beta,$$

where $\mu_{\boldsymbol{z}} \in \mathbb{R}$ and $\sigma_{\boldsymbol{z}}^2 \in \mathbb{R}$ are the mean and variance, and $\gamma, \beta \in \mathbb{R}$ are learnable affine parameters. The Lipschitz constant of this operation is upper-bounded as $L_{\mathrm{GN}} \leq \frac{|\gamma|}{\sqrt{\epsilon}}$ (see Lemma B.1). Since ϵ is a very small constant (e.g., 10^{-5}) to prevent division by zero, the Lipschitz constant can become excessively large in the worst-case scenario where the variance approaches zero. This potential for an explosive Lipschitz constant poses a significant challenge to the stability of the fixed-point iteration. To mitigate this issue, our Lipschitz MDEQ adopts a simpler normalization scheme, Mean-Only Group Normalization (MGN). This operation only centers the features by subtracting the mean, omitting the division by the standard deviation:

$$\mathrm{MGN}(\boldsymbol{z})_i = \gamma(z_i - \mu_{\boldsymbol{z}}) + \beta.$$

This seemingly small modification has a profound impact on the Lipschitz constant. The upper bound for

MGN is given by $L_{\mathrm{MGN}} \leq |\gamma|$ (see Lemma B.2). By removing the problematic $1/\sqrt{\epsilon}$ term, the Lipschitz constant becomes directly dependent only on the learnable parameter γ, making it significantly smaller and far more controllable. We treat the upper bound $\bar{\gamma}$ on the absolute value of γ as another adjustable hyperparameter, enabling precise control over the Lipschitz contribution from the normalization layer. In the implementation, the use of learnable parameters γ and β is optional. In the MDEQ implementation, the GN appearing in the residual block and fusion layer uses this option, and Lipschitz MDEQ adopts this approach. When these are not used, $\gamma = 1$ and $\beta = 0$ are set and therefore L_{MGN} is always 1. When using them, L_{MGN} can be controlled by arbitrarily determining $\bar{\gamma}$.

Activation Function MDEQ employs the ReLU function as its activation function. The ReLU function operates on each element of the input vector $\boldsymbol{z}$ and is defined as $\mathrm{ReLU}(\boldsymbol{z})_i := \max\{0, z_i\}$. The Lipschitz constant for this operation is clearly 1 (see Lemma B.3). To contribute to keeping the overall Lipschitz constant small, our Lipschitz MDEQ adopts the Scaled-ReLU (SReLU) function [Ali et al., 2020]. SReLU is defined as $\mathrm{SReLU}(\boldsymbol{z})_i := \max\{0, az_i\}$, where $a \in (0, 1]$ is a hyperparameter and the Lipschitz constant becomes $L_{\mathrm{SReLU}} = a$ (see Lemma B.4).

Dropout The dropout [Srivastava et al., 2014] is an important technique for suppressing overfitting and improving generalization performance. MDEQ employs variational dropout [Gal and Ghahramani, 2016] defined by $\mathrm{Drop}(\boldsymbol{z}) := \frac{1}{1-p}\boldsymbol{m} \odot \boldsymbol{z}$, where $p \in (0, 1)$ is the dropout rate and $\boldsymbol{m} \sim \mathrm{Bernoulli}(p)$ is the mask. The Lipschitz constant for this operation is $L_{\mathrm{Drop}} = 1/(1-p)$ (see Lemma B.5). In practice, dropout rate p is typically a small value ranging from 0 to 0.3, so the Lipschitz constant for the dropout operation does not become excessively large. Therefore, our Lipschitz MDEQ also adopts variational dropout.

Convolution A standard convolution layer performs an affine transformation on the input feature map $\boldsymbol{z}$. Conceptually, this operation can be expressed as $\mathrm{Conv}(\boldsymbol{z}) = W\boldsymbol{z} + \boldsymbol{b}$, where W is a linear operator derived from the convolutional kernel and $\boldsymbol{b}$ is a bias term. The Lipschitz constant for this operation is $L_{\mathrm{Conv}} = \|W\|_2$ (see Lemma B.6). In our Lipschitz MDEQ, we directly control this spectral norm to regulate the Lipschitz constant of the convolutional layers. Specifically, we enforce a constraint $\|W\|_2 \leq c$ by reprojecting the weights after each optimization step, where the upper bound $c > 0$ is a tunable hyperparameter we refer to as the target norm. We denote constrained Conv as $\mathrm{Conv}^\star$, and therefore $L_{\mathrm{Conv}^\star} = c$.

Residual Connection In the residual block of MDEQ, a residual connection is observed where the inputs are simply added together, such as $h(x) = x + g(x)$. This is an extremely effective mechanism introduced in [He et al., 2016]. However, since the Lipschitz constant of the identity mapping is 1, $L_h = 1 + L_g$ according to Lemma 3.1, contributing to an overall increase in the Lipschitz constant. The simplest approach to mitigate this is to use a convex combination for $\alpha \in (0, 1)$, such as $h'(x) = (1-\alpha)x + \alpha g(x)$. Then, we have $L_{h'} = (1-\alpha) + \alpha L_g$, achieving a smaller Lipschitz constant than the simple residual connection. Therefore, in Lipschitz MDEQ, all residual connections are made convex combinations, and the coefficient parameters are treated as hyperparameters. The convex combination parameter in the residual block is denoted by α_1 (Eq. (L1)). It should be noted that this convex combinations structure has been used in highway networks [Srivastava et al., 2015a, Srivastava et al., 2015b] and has been widely adopted as the fundamental form of gate mechanisms in gated recurrent units (GRUs) [Chung et al., 2014] and recent gated linear RNNs [Qin et al., 2023, Feng et al., 2024, Gu and Dao, 2024].

Downsample In the fusion layer of MDEQ, when passing states from high resolution to low resolution, a convolution operation is applied that is proportional to the difference in resolution (the blue arrow in Figure 1). Specifically, if $h_1(\boldsymbol{z}) := \mathrm{GN}(\mathrm{CONV}(\boldsymbol{z}))$ and $h_2(\boldsymbol{z}) := \mathrm{ReLU}(\mathrm{GN}(\mathrm{CONV}(\boldsymbol{z})))$, the operation $\mathrm{D}_{i,j}$ is defined as $\mathrm{D}_{i,j} := h_1 \circ h_2^{i-j-1}$. Therefore, the Lipschitz constant for this operation is $L_{\mathrm{Down}}^{i,j} = L_{h_1} L_{h_2}^{i-j-1}$, which depends on the Lipschitz constants of the activation function, normalization, and convolution operations. Our Lipschitz MDEQ implements the MDEQ downsampling operation without altering its form. Therefore, in Lipschitz MDEQ, the Lipschitz constant for this operation is $L_{\mathrm{Down}}^{i,j} = L_{\mathrm{MGN}} L_{\mathrm{Conv}^\star} (L_{\mathrm{SReLU}} L_{\mathrm{MGN}} L_{\mathrm{Conv}^\star})^{i-j-1}$.

Upsample In the fusion layer of MDEQ, when passing states from low resolution to high resolution, missing dimensions are filled using nearest-neighbor interpolation (the yellow arrow in Figure 1). This operation $\mathrm{U}_{i,j}$ takes a low-resolution input $\boldsymbol{z}_j \in \mathbb{R}^{H \times W}$ and outputs high-resolution $\boldsymbol{z}_{j \to i} \in \mathbb{R}^{sH \times tW}$, where $s, t \in \mathbb{N}$ are the vertical and horizontal scale factors, respectively. In MDEQ, both scale factors are defined as $2^{\mathrm{level_diff}}$, where level_diff is the difference between the indices of the hidden states; for example, level_diff between $\boldsymbol{z}_1$ and $\boldsymbol{z}_3$ is 2. The Lipschitz constant for this operation is $L_{\mathrm{up}} = \sqrt{st}$, i.e., $2^{\mathrm{level_diff}}$ (see Lemma B.7). Practically, the number of resolutions n is chosen as either 2 or 4, so the Lipschitz constant reaches a

maximum of $2^3 = 8$, which is a factor causing the overall Lipschitz constant to explode. Lipschitz MDEQ inherits this operation as well.

Multiscale Fusion In the fusion layer of MDEQ, $\tilde{\boldsymbol{z}}_i$ is the sum of itself $\hat{\boldsymbol{z}}_i$ and all $\hat{\boldsymbol{z}}_j$ $(i \neq j)$ that have undergone downsample and upsample (see Eq. (M2)). However, since the Lipschitz constant for the upsampling operation in particular can become large depending on level_diff, taking the sum may cause the Lipschitz constant to explode in the fusion layer. Therefore, Lipschitz MDEQ first performs a convex combination between itself $\hat{\boldsymbol{z}}_i$ and the other upsampled or downsampled resolutions $\hat{\boldsymbol{z}}_j$ $(i \neq j)$. Then, for upsampling and downsampling, it takes a weighted average, applying smaller weights as level_diff increases (see also Eq. (L2)), i.e.,

$$\tilde{\boldsymbol{z}}_i = (1-\alpha_2)\hat{\boldsymbol{z}}_i + \alpha_2 \left(\sum_{j \neq i} w_{ij} \mathrm{Fuse}_{i,j}(\hat{\boldsymbol{z}}_i) \right),$$

where $\alpha_2 \in (0,1)$, $\mathrm{Fuse}_{i,j}$ is $\mathrm{D}_{i,j}$ when $j < i$ and $\mathrm{U}_{i,j}$ when $j > i$, and the weights w_{ij} are computed dynamically via a softmax function over penalties proportional to the level_diff of the upsampling paths:

$$w_{ij} = \frac{\exp(-p_{ij})}{\sum_{k \neq i} \exp(-p_{ik})}, \text{where } p_{ij} = \begin{cases} j - i & \text{if } j > i \\ 0 & \text{if } j < i \end{cases}.$$

Crucially, the weights w_{ij} are designed to penalize contributions from distant branches. This mechanism ensures that upsampling operations with large level_diff, and thus large Lipschitz constants, are assigned exponentially smaller weights. By adaptively down-weighting these potentially explosive terms, our fusion layer effectively suppresses the Lipschitz constant, contributing significantly to the overall stability.

3.3 Lipschitz Constant of Lipschitz MDEQ

The hidden state of MDEQ consists of a tuple of n feature maps, $\boldsymbol{z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n]$, where each $\boldsymbol{z}_i$ can be viewed as a vector in a d_i-dimensional Euclidean space, $\mathbb{R}^{d_i}$. The full hidden state $\boldsymbol{z}$ therefore resides in the product space $\mathcal{Z} = \mathbb{R}^{d_1} \times \cdots \times \mathbb{R}^{d_n}$. The mapping $f_{\boldsymbol{\theta}}$ takes a tuple of hidden states $\boldsymbol{z} \in \mathcal{Z}$ and outputs $\bar{\boldsymbol{z}} = f_{\boldsymbol{\theta}}(\boldsymbol{z}) \in \mathcal{Z}$. We analyze the Lipschitz constant of $f_{\boldsymbol{\theta}}$ with respect to the norm on $\mathcal{Z}$ defined by

$$\|\boldsymbol{z}\|_{\mathcal{Z}} := \sqrt{\sum_{i=1}^{n} \|\boldsymbol{z}_i\|_2^2},$$

where $\|\cdot\|_2$ denotes the standard Euclidean norm. This norm is equivalent to the L_2-norm of the vector formed by concatenating all hidden states $\boldsymbol{z}_i \in \mathbb{R}^{d_i}$, and naturally satisfies the axioms of a norm.

Residual Block and Post-Fusion Layer In the residual block, the branch processing function $\hat{f} \colon \mathcal{Z} \to \mathcal{Z}$ acts on each resolution branch independently. We can thus decompose it into n component functions $\hat{f}_i : \mathbb{R}^{d_i} \to \mathbb{R}^{d_i}$, such that for an input $\boldsymbol{z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n]$, the output is simply $\hat{f}(\boldsymbol{z}) = [\hat{f}_1(\boldsymbol{z}_1), \ldots, \hat{f}_n(\boldsymbol{z}_n)]$. Note that although these functions differ in their input and output dimensions, the modules constituting them are identical. That is, $\hat{f}_i$ possesses the same Lipschitz constant for any $i \in [n]$, which we denote as $\hat{L}$. Then, we have

$$\begin{aligned} \|\hat{f}(\boldsymbol{z}) - \hat{f}(\boldsymbol{z}')\|_{\mathcal{Z}}^2 &= \sum_{i \in [n]} \|\hat{f}_i(\boldsymbol{z}_i) - \hat{f}_i(\boldsymbol{z}_i')\|_2^2 \\ &\leq \sum_{i \in [n]} \hat{L}^2 \|\boldsymbol{z}_i - \boldsymbol{z}_i'\|_2^2 \\ &= \hat{L}^2 \|\boldsymbol{z} - \boldsymbol{z}'\|_{\mathcal{Z}}^2. \end{aligned}$$

Therefore, $\hat{f}$ also has the same Lipschitz constant, $\hat{L}$. From the definition of $\hat{f}_i$ (see Eq. (L1)) and Lemma 3.1, it can be shown that

$$\begin{aligned} \hat{L} = (1-\alpha_1) L_{\mathrm{MGN}} L_{\mathrm{SReLU}} \\ + \alpha_1 L_{\mathrm{MGN}}^3 L_{\mathrm{Conv}^\star}^2 L_{\mathrm{SReLU}}^2 L_{\mathrm{Drop}}. \end{aligned} \tag{5}$$

Similarly, since the processing for each branch in the post-fusion layer $\bar{f}$ is completely independent, let the processing for each branch be denoted by $\bar{f}_i$ and the Lipschitz constant for these be $\bar{L}$. Then, the Lipschitz constant for $\bar{f}$ is also $\bar{L}$. From the definition of $\bar{f}_i$ (see Eq. (L3)), we have

$$\bar{L} = L_{\mathrm{MGN}} L_{\mathrm{Conv}^\star} L_{\mathrm{SReLU}}. \tag{6}$$

Fusion Layer In the fusion layer, the processing of each branch is not independent, as the hidden states influence each other. Therefore, we define the mapping $\tilde{f}_i \colon \mathcal{Z} \to \mathbb{R}^{d_i}$ to take a tuple of n hidden states $\hat{\boldsymbol{z}} \in \mathcal{Z}$ as input and output a single hidden state $\tilde{\boldsymbol{z}}_i \in \mathbb{R}^{d_i}$, i.e., $\tilde{f}(\hat{\boldsymbol{z}}) = [\tilde{f}_1(\hat{\boldsymbol{z}}), \ldots, \tilde{f}_n(\hat{\boldsymbol{z}})]$. Suppose $\tilde{f}_i$ is $\tilde{L}_i$-Lipschitz $(i \in [n])$, i.e., $\|\tilde{f}_i(\hat{\boldsymbol{z}}) - \tilde{f}_i(\hat{\boldsymbol{z}}')\|_2 \leq \tilde{L}_i \|\hat{\boldsymbol{z}} - \hat{\boldsymbol{z}}'\|_{\mathcal{Z}}$. Then, we have

$$\begin{aligned} \|\tilde{f}(\hat{\boldsymbol{z}}) - \tilde{f}(\hat{\boldsymbol{z}}')\|_{\mathcal{Z}}^2 &= \sum_{i \in [n]} \|\tilde{f}_i(\hat{\boldsymbol{z}}) - \tilde{f}_i(\hat{\boldsymbol{z}}')\|_2^2 \\ &\leq \sum_{i \in [n]} \tilde{L}_i^2 \|\hat{\boldsymbol{z}} - \hat{\boldsymbol{z}}'\|_{\mathcal{Z}}^2. \end{aligned}$$

Therefore, $\tilde{f}$ is $\sqrt{\sum_{i \in [n]} \tilde{L}_i^2}$-Lipschitz. From the definition of $\tilde{f}_i$ (see Eq. (L2)), we have

$$\tilde{L}_i = \sqrt{(1-\alpha_2)^2 + \alpha_2^2 \sum_{j \neq i} (w_{ij} L_{\mathrm{fuse}}^{i,j})^2}. \tag{7}$$

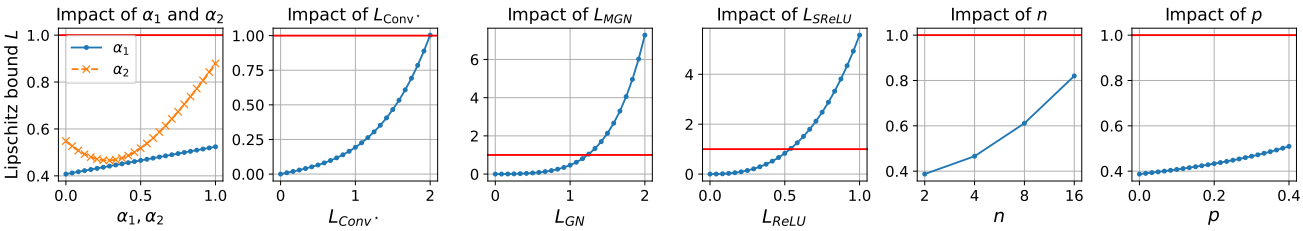


Figure 2: Effect of hyperparameters constituting $f_{\boldsymbol{\theta}}$ on Lipschitz bound L. Each graph is plotted by varying only one parameter while holding all others constant. Fixed parameters are set as $\alpha_1 = 0.5, \alpha_2 = 0.3, L_{\mathrm{Conv}^\star} = 1.5, L_{\mathrm{MGN}} = 1.0, L_{\mathrm{SReLU}} = 0.4, n = 4$, and $p = 0.3$. Note that changing the values of the fixed parameters will also change the plot.

Since $f_{\boldsymbol{\theta}}$ is the composite mapping of these three blocks, its Lipschitz constant is determined by their product and is equal to

$$L = \hat{L}\sqrt{\sum_{i\in[n]} \tilde{L}_i^2}\bar{L}, \tag{8}$$

where $\hat{L}, \bar{L}, \tilde{L}_i$ are defined as in (5), (6), (7), respectively.

Sensitivity of Each Hyperparameter The Lipschitz constant for the forward pass is determined by the Lipschitz constants of each operation constituting $f_{\boldsymbol{\theta}}$, as given by equation (8), and the Lipschitz constant of each operation is a hyperparameter. Specifically, it is determined by the two convex combination parameters α_1 and α_2, the target norm $L_{\mathrm{Conv}^\star} = c$ in the convolution operation, the affine parameter $L_{\mathrm{MGN}} = \bar{\gamma}$ in the normalization operation, the slope $L_{\mathrm{SReLU}} = a$ of the activation function, the number of resolution branches n, and the dropout rate p. Figure 2 plots how much each parameter affects the Lipschitz constant L for the entire forward pass. The magnitude of the impact depends on the order of the Lipschitz constant for each operation appearing within L, and it is evident that L is particularly strongly influenced by $L_{\mathrm{SReLU}}, L_{\mathrm{MGN}}$, and $L_{\mathrm{Conv}^\star}$. Furthermore, especially when these values are large, the small convex combination parameters α_1, α_2 play a crucial role in keeping L small. For example, when $L_{\mathrm{SReLU}} = 0.35, L_{\mathrm{Conv}^\star} = 2.0, L_{\mathrm{MGN}} = 1.0, \alpha_1 = 0.5, \alpha_2 = 0.3, n = 4$, and $p = 0.3$, the Lipschitz constant for the forward pass is $L = 0.86$, satisfying the contraction property.

3.4 Side Effects in Backward Pass

As described already, we restructured the model architecture so that the fixed-point mapping $f_{\boldsymbol{\theta}}$ appearing in the forward pass (1) becomes a contraction mapping; i.e., we ensured that the Lipschitz constant $L = \sup_{\boldsymbol{z}\in\mathcal{Z}} \|J_{f_{\boldsymbol{\theta}}}(\boldsymbol{z})\|_2$ is less than 1, where $J_{f_{\boldsymbol{\theta}}}$ is the Jacobian matrix of $f_{\boldsymbol{\theta}}$. This guarantees fixed-point convergence in the forward pass and is also expected to accelerate the forward pass. Meanwhile, beneficial effects are also observed in the backward pass. The Lipschitz constant for the fixed-point mapping appearing in the backward pass (4) is $\|J_{f_{\boldsymbol{\theta}}}(\boldsymbol{z}^\star)\|_2$ (see Lemma B.8), where $\boldsymbol{z}^\star \in \mathcal{Z}$ is the fixed point obtained in the forward pass. Since $\|J_{f_{\boldsymbol{\theta}}}(\boldsymbol{z}^\star)\|_2 \leq \sup_{\boldsymbol{z}\in\mathcal{Z}} \|J_{f_{\boldsymbol{\theta}}}(\boldsymbol{z})\|_2$, the Lipschitz constant for the backward pass is always smaller than that in the forward pass. Therefore, forcing the forward pass Lipschitz constant to be less than 1 also means suppressing the backward pass Lipschitz constant to be less than 1. This guarantees fixed-point convergence for the backward pass as well, and acceleration of the backward pass is also anticipated.

4 NUMERICAL EXPERIMENTS

For details on the experimental environment and hyperparameters, see Appendix C.1. In all experiments, we control the Lipschitz constant L by fixing $L_{\mathrm{MGN}} = 1.0$, $L_{\mathrm{Conv}^\star} = 2.0$, $\alpha_1 = 0.5$, $\alpha_2 = 0.3$, $n = 4$, and $p = 0.3$ and varying the $L_{\mathrm{SReLU}} = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. Specifically, $L = \{0.03, 1.0, 14.43\}$ are obtained when $L_{\mathrm{SReLU}} = \{0.1, 0.4, 1.0\}$, respectively.

Comparison of Fixed-point Convergence In Lipschitz MDEQ, forcing the Lipschitz constant in the forward pass to be less than 1 guarantees the existence of a unique fixed point. Figure 3 plots fixed-point convergence for CIFAR-10 classification. Number of fixed-point iterations is fixed at 18 for the forward pass and 20 for the backward pass. The evaluation metric is relative residual $\|f_{\boldsymbol{\theta}}(\boldsymbol{z};\boldsymbol{x}) - \boldsymbol{z}\|/\|f_{\boldsymbol{\theta}}(\boldsymbol{z};\boldsymbol{x})\|$. Throughout training, Lipschitz MDEQ achieves overwhelming fixed-point convergence, demonstrating better convergence than existing methods even in cases where convergence is not theoretically guaranteed (i.e., $L \geq 1$). We obtained similar results for the forward pass during testing (see Appendix C.2).

Acceleration in Image Classification Tasks Table 1 summarizes the results of MDEQ and several acceleration methods for CIFAR-10 classification, along with our Lipschitz MDEQ. All fixed-point iteration

Table 1: Comparison of different acceleration techniques for MDEQ on CIFAR-10. We report top-1 accuracy, total training speed, and average number of fixed-point iterations (NFEs) and runtime (Time) for the forward and backward passes. Best results in each column are highlighted in **bold**. JR: Jacobian regularization [Bai et al., 2021]; PG: phantom gradient [Geng et al., 2021]; and JFB: Jacobian-free backpropagation [Fung et al., 2022]. †Only JR has the maximum NFEs set to 7 and 8.

Model and method	Size	Speed-up	Acc.	Train forward		Train backward		Test forward	
				NFEs	Time (ms)	NFEs	Time (ms)	NFEs	Time (ms)
MDEQ	10M	1.0×	93.23	17.9	217	20	291	18.0	63.9
MDEQ + JR	10M	2.22×	91.86	7.0†	74.8	8.0†	97.8	7.0†	39.2
MDEQ + PG	10M	1.41×	92.92	15.6	220	–	87.6	18.0	63.9
MDEQ + JFB	10M	2.84×	88.13	12.3	142	–	**30.6**	11.3	44.3
Lipschitz MDEQ ($L = 14.43$)	10M	1.28×	**93.73**	16.0	187	19.9	193	15.5	54.1
Lipschitz MDEQ ($L = 1.0$)	10M	3.33×	91.44	5.4	75.7	2.9	52.9	4.8	22.6
Lipschitz MDEQ ($L = 0.03$)	10M	**4.75**×	90.47	**2.0**	**44.0**	**2.0**	42.1	**2.0**	**13.3**
Lipschitz MDEQ ($L = 14.43$) + JFB	10M	3.66×	88.66	7.6	90.18	-	24.63	7.11	26.51
Lipschitz MDEQ ($L = 1.0$) + JFB	10M	4.98×	87.53	4.2	55.34	-	24.64	4.0	17.57
Lipschitz MDEQ ($L = 0.03$) + JFB	10M	6.43×	87.26	2.0	34.98	-	24.67	2.0	11.3

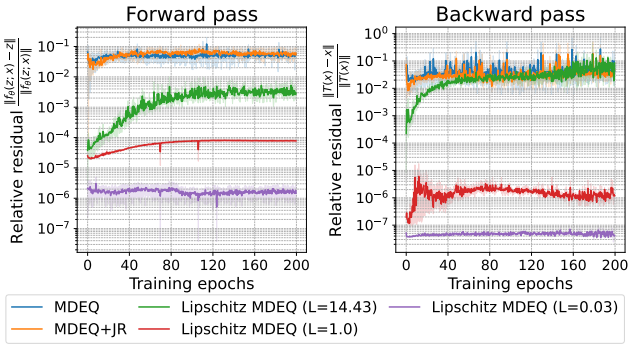


Figure 3: Comparison of fixed-point convergence in forward and backward passes of training.

evaluation metrics used residual, with a threshold of 1e-3. The maximum iteration was set to 18 for the forward pass and 20 for the backward pass. In both the forward pass during training and testing and the backward pass during training, Lipschitz MDEQ achieves significantly faster fixed-point convergence than existing methods, reducing runtime. However, it was observed that the stricter the contractive constraints, the more test accuracy deteriorated. Furthermore, in theory, the smaller the Lipschitz constant, the better JFB works. Therefore, we also report the results of combining JFB with the Lipschitz MDEQ in Table 1. This combination unlocks further speed improvements, but compared to Lipschitz MDEQ without JFB, the deterioration in accuracy is even more pronounced.

Trade-off between Accuracy and Runtime Figure 4 plots accuracy and speed as the Lipschitz constant is varied. The results shown demonstrate that controlling the Lipschitz constant allows the trade-off between accuracy and speed to be managed. Furthermore, even when theoretical guarantees cannot be obtained ($L \geq 1$), sufficient acceleration equivalent to or exceeding existing methods was achieved. In addition, by setting the Lipschitz constant to a large value without theoretical guarantees, our Lipschitz MDEQ achieves test accuracy equivalent to or exceeding that of MDEQ, along with a slight increase in speed.

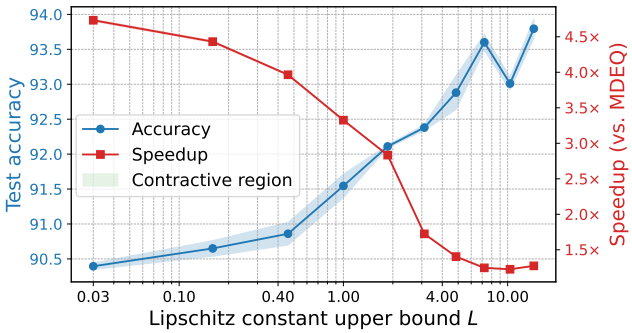


Figure 4: Accuracy and speed trade-off in Lipschitz MDEQ. The region where convergence is guaranteed, $L < 1$, is shaded green.

4.1 Ablation Study and Discussion

By systematically removing each modification applied to the individual operations that constitute Lipschitz MDEQ, we aim to verify the contribution and importance of each technique toward achieving both accuracy and speed. Table 2 summarizes the experimental results obtained under settings where only the modification of a single operation was undone. The default parameters inherit the values of Lipschitz MDEQ with $L = 1.0$. The settings are as follows. (S1) We removed the constraint $\bar{\gamma} \leq 1$ on the affine parameter of the normalization operation MGN. (S2) We changed the normalization operation from MGN to GN. (S3) We changed the convolution operation from Conv* to general Conv. (S4) We removed the weighted averaging mechanism via the softmax function in the fusion layer and replaced it with simple summation. (S5) We removed the convex combination in the residual block

Table 2: Ablation study for Lipschitz MDEQ on CIFAR-10. We report top-1 accuracy, total training speed, and average number of fixed-point iterations (NFEs) and runtime (Time) for the forward and backward passes. Best results in each column are highlighted in **bold**.

Model and method	Size	Speed-up	Acc.	Train forward		Train backward		Test forward	
				NFEs	Time (ms)	NFEs	Time (ms)	NFEs	Time (ms)
MDEQ	10M	1.0×	93.23	17.9	217	20	291	18.0	63.9
Lipschitz MDEQ ($L = 1.0$)	10M	3.33×	91.44	5.4	75.7	2.9	52.9	4.8	22.6
(S1) Lipschitz MDEQ - MGN clip	10M	2.81×	91.99	6.4	90.5	4.7	65.3	5.7	25.2
(S2) Lipschitz MDEQ - MGN	10M	0.95×	92.78	18.0	225	20.0	296	18.0	72.9
(S3) Lipschitz MDEQ - Conv*	10M	**3.42**×	91.94	**3.8**	**61.4**	5.6	73.0	**3.7**	**19.0**
(S4) Lipschitz MDEQ - softmax	10M	3.41×	91.29	5.4	76.5	**2.8**	**48.2**	4.9	20.6
(S5) Lipschitz MDEQ - α_1	10M	2.72×	92.35	6.6	90.9	5.5	71.9	6.0	25.9
(S6) Lipschitz MDEQ - α_2	10M	1.64×	92.73	9.7	120	18.9	179	9.8	33.2
(S7) Lipschitz MDEQ - α_1, α_2	10M	1.37×	**93.35**	14.5	169	19.6	184	14.5	45.4

and replaced it with pure residual connection. (S6) We removed the convex combination in the fusion layer and replaced it with pure residual connection. (S7) We removed the convex combinations in both the residual block and fusion layer and replaced both combinations with pure residual connections.

Table 2 shows that the most impactful modification in the transition from MDEQ to Lipschitz MDEQ was changing the normalization operation from GN to MGN (see (S2)). In (S2), despite techniques other than normalization still attempting to maintain a small overall Lipschitz constant, fixed-point convergence deteriorates significantly, resulting in the longest runtime. This result indicates that our modification is functioning as intended, and therefore we recommend using MGN rather than GN from the perspective of the Lipschitz constant of the normalization operation.

Furthermore, the results in (S5)–(S7) demonstrate that simple residual connections lead to worse fixed-point convergence compared to convex combinations. Similarly, while this result validates our theory, note that using convex combinations results in a deterioration of accuracy. There may be room for improvement in achieving high test accuracy and speed-up through superior fixed-point convergence.

In addition, the results of (S3) and (S4) could not be predicted by our theoretical analysis. First, the results of (S4) show that while taking a weighted average using the softmax function in the fusion layer undoubtedly contributes to theoretically suppressing the overall Lipschitz constant, the experimental benefits were limited. Next, (S3) demonstrates that imposing no constraints on the Lipschitz constant of the convolution operation does not worsen fixed-point convergence; rather, it partially improves it. Moreover, since parameter norm constraints naturally reduce a model's expressive power, (S3) achieves higher test accuracy than the full-specification Lipschitz MDEQ (second row in the table). Therefore, we must conclude that the hyperparameter setting for Lipschitz MDEQ that removes only the constraint on the convolution operation (S3) is practical in terms of both test accuracy and runtime. Of course, it is important to note that the acceleration of fixed-point convergence in (S3) is not fully guaranteed theoretically. Thus, Lipschitz MDEQ still exhibits several gaps between theory and experimental observations, pointing to one direction for future work.

Finally, to accelerate DEQ based on our approach, theoretically any modification is acceptable as long as the Lipschitz constant can be kept small; our model modification is merely one example. While this study has demonstrated the effectiveness of this approach, important future work will also involve exploring model modifications that suppress the Lipschitz constant without compromising test accuracy, as well as modifications effective for other explicit models.

5 CONCLUSION

This paper proposes Lipschitz MDEQ, a modified architecture of MDEQ. The proposed model allows users to determine the Lipschitz constant of the fixed-point mapping in the forward pass, which can thereby be made into a contracting mapping, guaranteeing fixed-point convergence. This dramatically improves fixed-point convergence in the forward pass. Furthermore, since the Lipschitz constant of the fixed-point mapping in the backward pass is smaller than that in the forward pass, the acceleration of fixed-point convergence extends to the backward pass as well. Consequently, the proposed architecture can dramatically reduce the time required for training and inference compared to existing models and acceleration techniques, albeit at the cost of slightly lower accuracy. We are confident that our results represent a significant step toward developing high-performance, fast models with superior memory efficiency, addressing the critical drawback of the low learning and inference speeds in DEQ.

Acknowledgements

We are sincerely grateful to Program Chairs, Area Chairs, and three anonymous reviewers for helping us improve the original manuscript. We would like to thank Junya Morioka for his technical support in the numerical experiments. This work was supported by the Japan Society for the Promotion of Science (JSPS) KAKENHI Grant Number 24K14846 awarded to Hideaki Iiduka.

References

[Agarwala and Schoenholz, 2022] Agarwala, A. and Schoenholz, S. S. (2022). Deep equilibrium networks are sensitive to initialization statistics. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 136–160.

[Agrawal et al., 2019] Agrawal, A., Amos, B., Barratt, S. T., Boyd, S. P., Diamond, S., and Kolter, J. Z. (2019). Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, volume 32, pages 9558–9570.

[Ali et al., 2020] Ali, R. E., So, J., and Avestimehr, A. S. (2020). On polynomial approximations for privacy-preserving and verifiable ReLU networks. In *Neural Information Processing Systems Workshop: Privacy Preserving Machine Learning*.

[Almeida, 1987] Almeida, L. B. (1987). A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *Proceedings of the IEEE First International Conference on Neural Networks*, volume 2, pages 609–618.

[Amos and Kolter, 2017] Amos, B. and Kolter, J. Z. (2017). OptNet: differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 136–145.

[Anderson, 1965] Anderson, D. G. (1965). Iterative procedures for nonlinear integral equations. *Journal of the ACM*, 12(4):547–560.

[Anil et al., 2019] Anil, C., Lucas, J., and Grosse, R. B. (2019). Sorting out lipschitz function approximation. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 291–301.

[Bai et al., 2022a] Bai, S., Geng, Z., Savani, Y., and Kolter, J. Z. (2022a). Deep equilibrium optical flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 620–630.

[Bai et al., 2019a] Bai, S., Kolter, J. Z., and Koltun, V. (2019a). Deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 32, pages 688–699.

[Bai et al., 2019b] Bai, S., Kolter, J. Z., and Koltun, V. (2019b). Trellis networks for sequence modeling. In *International Conference on Representation Learning*.

[Bai et al., 2020] Bai, S., Koltun, V., and Kolter, J. Z. (2020). Multiscale deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 33, pages 5238–5250.

[Bai et al., 2021] Bai, S., Koltun, V., and Kolter, J. Z. (2021). Stabilizing equilibrium models by jacobian regularization. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 554–565.

[Bai et al., 2022b] Bai, S., Koltun, V., and Kolter, J. Z. (2022b). Neural deep equilibrium solvers. In *International Conference on Representation Learning*.

[Bai and Melas-Kyriazi, 2024] Bai, X. and Melas-Kyriazi, L. (2024). Fixed point diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9430–9440.

[Banach, 1922] Banach, S. (1922). Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae*, 3(1):133–181.

[Béthune et al., 2022] Béthune, L., Boissin, T., Serrurier, M., Mamalet, F., Friedrich, C., and Gonzalez Sanz, A. (2022). Pay attention to your loss: understanding misconceptions about 1-lipschitz neural networks. In *Advances in Neural Information Processing Systems*, volume 35, pages 20077–20091.

[Broyden, 1965] Broyden, C. G. (1965). A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation*, 19(92):577–593.

[Bubba et al., 2025] Bubba, T. A., Santacesaria, M., and Sebastiani, A. (2025). TomoSelfDEQ: self-supervised deep equilibrium learning for sparse-angle CT reconstruction. In *Scale Space and Variational Methods in Computer Vision*, pages 334–346.

[Burger et al., 2025] Burger, A., Thiede, L., Aspuru-Guzik, A., and Vijaykumar, N. (2025). DEQuify your force field: More efficient simulations using deep equilibrium models. In *AI for Accelerated Materials Design - ICLR 2025 Workshop*.

[Cao et al., 2024] Cao, J., Shi, Y., Zhang, K., Zhang, Y., Timofte, R., and Gool, L. V. (2024). Deep equilibrium diffusion restoration with parallel sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2824–2834.

[Chandler et al.,] Chandler, E. P., Shoushtari, S., Liu, J., Asif, M. S., and Kamilov, U. S. Overcoming distribution shifts in plug-and-play methods with test-time training. In *2023 IEEE 9th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 186–190.

[Chen et al., 2023] Chen, Q., Wang, Y., Geng, Z., Wang, Y., Yang, J., and Lin, Z. (2023). Equilibrium image denoising with implicit differentiation. *IEEE Transactions on Image Processing*, 32:1868–1881.

[Chen et al., 2022] Chen, Q., Wang, Y., Wang, Y., Yang, J., and Lin, Z. (2022). Optimization-induced graph implicit nonlinear diffusion. In *Proceedings of the 38th International Conference on Machine Learning*, volume 162, pages 3648–3661.

[Chen et al., 2018] Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, pages 6572–6583.

[Chen et al., 2021] Chen, T., Lasserre, J. B., Magron, V., and Pauwels, E. (2021). Semialgebraic representation of monotone deep equilibrium models and applications to certification. In *Advances in Neural Information Processing Systems*, volume 34, pages 27146–27159.

[Choi et al., 2020] Choi, Y., Uh, Y., Yoo, J., and Ha, J.-W. (2020). StarGAN v2: Diverse image synthesis for multiple domains. In *Proceeedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8185–8194.

[Chu et al., 2024] Chu, H., Wei, S., Liu, T., Zhao, Y., and Miyatake, Y. (2024). Lyapunov-stable deep equilibrium models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11615–11623.

[Chung et al., 2014] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NeurIPS 2014 Deep Learning and Representation Learning Workshop*, volume https://arxiv.org/abs/1412.3555.

[Cisse et al., 2017] Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N. (2017). Parseval networks: Improving robustness to adversarial examples. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 854–863.

[Dandapanthula and Ramdas, 2025] Dandapanthula, S. and Ramdas, A. (2025). Gradient flow for deep equilibrium single-index models. *arXiv*, https://arxiv.org/abs/2511.16976.

[Daniele et al., 2025] Daniele, C., Villa, S., Vaiter, S., and Calatroni, L. (2025). Deep equilibrium models for poisson imaging inverse problems via mirror descent. *arXiv*, https://arxiv.org/abs/2507.11461.

[de Avila Belbute-Peres et al., 2018] de Avila Belbute-Peres, F., Smith, K. A., Allen, K. R., Tenenbaum, J., and Kolter, J. Z. (2018). End-to-end differentiable physics for learning and control. In *Advances in Neural Information Processing Systems*, volume 31, pages 7178–7189.

[Dehghani et al., 2019] Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, L. (2019). Universal transformers. In *International Conference on Representation Learning*.

[Ding et al., 2023] Ding, S., Cui, T., Wang, J., and Shi, Y. (2023). Two sides of the same coin: Bridging deep equilibrium models and neural ODEs via homotopy continuation. In *Advances in Neural Information Processing Systems*, volume 36, pages 28053–28071.

[Djolonga and Krause, 2017] Djolonga, J. and Krause, A. (2017). Differentiable learning of submodular models. In *Advances in Neural Information Processing Systems*, volume 30, pages 1013–1023.

[Drucker and Le Cun, 1992] Drucker, H. and Le Cun, Y. (1992). Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks*, 3(6):991–997.

[Dupont et al., 2019] Dupont, E., Doucet, A., and Teh, Y. W. (2019). Augmented neural odes. In *Advances in Neural Information Processing Systems*, volume 32, pages 3140–3150.

[Ertenl et al., 2022] Ertenl, C. U., Akbas, E., and Cinbis, R. G. (2022). Streaming multiscale deep equilibrium models. In *Computer Vision – ECCV 2022*, volume 13671, pages 189–205.

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

[Feng et al., 2024] Feng, L., Tung, F., Ahmed, M. O., Bengio, Y., and Hajimirsadeghi, H. (2024). Were RNNs all we needed? *arXiv*, https://arxiv.org/abs/2410.01201.

[Feng and Kolter, 2023] Feng, Z. and Kolter, J. Z. (2023). On the neural tangent kernel of equilibrium models. *arXiv*, https://arxiv.org/abs/2310.14062.

[Fung et al., 2022] Fung, S. W., Heaton, H., Li, Q., McKenzie, D., Osher, S. J., and Yin, W. (2022). JFB: jacobian-free backpropagation for implicit networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6648–6656.

[Gabor et al., 2024] Gabor, M., Piotrowski, T., and Cavalcante, R. L. G. (2024). Positive concave deep equilibrium models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 14365–14381.

[Gal and Ghahramani, 2016] Gal, Y. and Ghahramani, Z. (2016). A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 29, pages 1019–1027.

[Gao et al., 2023] Gao, T., Huo, X., Liu, H., and Gao, H. (2023). Wide neural networks as gaussian processes: Lessons from deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 36, pages 54918–54951.

[Gao et al., 2022] Gao, T., Liu, H., Liu, J., Rajan, H., and Gao, H. (2022). A global convergence theory for deep relu implicit networks via overparameterization. In *International Conference on Representation Learning*.

[Gao et al., 2024] Gao, W., Hou, Z., Xu, H., and Liu, X. (2024). Certified robustness for deep equilibrium models via serialized random smoothing. In *Advances in Neural Information Processing Systems*, volume 38, pages 1370–1401.

[Geng and Kolter, 2023] Geng, Z. and Kolter, J. Z. (2023). TorchDEQ: a library for deep equilibrium models. https://github.com/locuslab/torchdeq.

[Geng et al., 2023] Geng, Z., Pokle, A., and Kolter, J. Z. (2023). One-step diffusion distillation via deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 36, pages 41914–41931.

[Geng et al., 2021] Geng, Z., Zhang, X.-Y., Bai, S., Wang, Y., and Lin, Z. (2021). On training implicit models. In *Advances in Neural Information Processing Systems*, volume 34, pages 24247–24260.

[Geuter et al., 2025] Geuter, J., Bonet, C., Korba, A., and Alvarez-Melis, D. (2025). DDEQs: distributional deep equilibrium models through wasserstein gradient flows. In *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, volume 258, pages 3988–3996.

[Ghaoui et al., 2021] Ghaoui, L. E., Gu, F., Travacca, B., Askari, A., and Tsai, A. (2021). Implicit deep learning. *SIAM Journal on Mathematics of Data Science*, 3(3):930–958.

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

[Gkillas et al., 2023] Gkillas, A., Ampeliotis, D., and Berberidis, K. (2023). Deep equilibrium models meet federated learning. In *2023 31st European Signal Processing Conference (EUSIPCO)*, pages 1873–1877.

[Goemaere et al., 2024] Goemaere, C., Deleu, J., and Demeester, T. (2024). Accelerating hopfield network dynamics: Beyond synchronous updates and forward euler. In *Proceedings of the 1st ECAI Workshop on "Machine Learning Meets Differential Equations: From Theory to Applications"*, volume 255, pages 1–21.

[Gould et al., 2022] Gould, S., Hartley, R., and Campbell, D. (2022). Deep declarative networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):3988–4004.

[Grathwohl et al., 2019] Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., and Duvenaud, D. (2019). FFJORD: free-form continuous dynamics for scalable reversible generative models. In *International Conference on Representation Learning*.

[Gu and Dao, 2024] Gu, A. and Dao, T. (2024). Mamba: Linear-time sequence modeling with selective state spaces. In *Proceedings of the Conference on Language Modeling*.

[Gu et al., 2020] Gu, F., Chang, H., Zhu, W., Sojoudi, S., and El Ghaoui, L. (2020). Implicit graph neural networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 11984–11995.

[Guan et al., 2024] Guan, P., Iqbal, N., Davenport, M. A., and Masood, M. (2024). Solving inverse problems with model mismatch using untrained neural networks within model-based architectures. *Transactions on Machine Learning Research*.

[Gulrajani et al., 2017] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, volume 30, pages 5767–5777.

[Güngör et al., 2024] Güngör, A., Askin, B., Soydan, D. A., Top, C. B., Saritas, E. U., and Çukur, T. (2024). DEQ-MPI: a deep equilibrium reconstruction with learned consistency for magnetic particle imaging. *IEEE Transactions on Medical Imaging*, 43(1):321–334.

[Gurumurthy et al., 2021] Gurumurthy, S., Bai, S., Manchester, Z., and Kolter, J. Z. (2021). Joint inference and input optimization in equilibrium networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 16818–16832.

[Havens et al., 2023] Havens, A. J., Araujo, A., Garg, S., Khorrami, F., and Hu, B. (2023). Exploiting connections between lipschitz structures for certifiably robust deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 36, pages 21658–21674.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[HU et al., 2024] HU, J., Gan, W., Sun, Z., An, H., and Kamilov, U. (2024). A Plug-and-Play image registration network. In *International Conference on Representation Learning*, pages 48980–49000.

[Hu et al., 2023] Hu, J., Shoushtari, S., Zou, Z., Liu, J., Sun, Z., and Kamilov, U. S. (2023). Robustness of deep equilibrium architectures to changes in the measurement mode. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5.

[Jafarpour et al., 2022] Jafarpour, S., Abate, M., Davydov, A., Bullo, F., and Coogan, S. (2022). Robustness certificates for implicit neural networks: A mixed monotone contractive approach. In *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, volume 168, pages 917–930.

[Jafarpour et al., 2021] Jafarpour, S., Davydov, A., Proskurnikov, A. V., and Bullo, F. (2021). Robust implicit networks via non-euclidean contractions. In *Advances in Neural Information Processing Systems*, volume 34, pages 9857–9868.

[Kawaguchi, 2021] Kawaguchi, K. (2021). On the theory of implicit deep learning: Global convergence with implicit layers. In *International Conference on Representation Learning*.

[Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Representation Learning*.

[Koyama et al., 2022] Koyama, Y., Murata, N., Uhlich, S., Fabbro, G., Takahashi, S., and Mitsufuji, Y. (2022). Music source separation with deep equilibrium models. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 296–300.

[Krantz and Parks, 2012] Krantz, S. G. and Parks, H. R. (2012). *The Implicit Function Theorem: History, Theory, and Applications*. Birkhäuser Boston.

[Kreyszig, 1978] Kreyszig, E. (1978). *Introductory Functional Analysis with Applications*. John Wiley & Sons.

[Krizhevsky, 2009] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[Li et al., 2022] Li, M., Wang, Y., and Lin, Z. (2022). CerDEQ: Certifiable deep equilibrium model. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 12998–13013.

[Liao et al., 2018] 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. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 3088–3097.

[Ling et al., 2024] Ling, Z., Li, L., Feng, Z., Zhang, Y., Zhou, F., Qiu, R. C., and Liao, Z. (2024). Deep equilibrium models are almost equivalent to not-so-deep explicit models for high-dimensional gaussian mixtures. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 30585–30609.

[Ling et al., 2023] Ling, Z., Xie, X., Wang, Q., Zhang, Z., and Lin, Z. (2023). Global convergence of over-parameterized deep equilibrium models. In *Proceed-*

ings of The 26th International Conference on Artificial Intelligence and Statistics, volume 206, pages 767–787.

[Liu et al., 2021] Liu, J., Kawaguchi, K., Hooi, B., Wang, Y., and Xiao, X. (2021). EIGNN: efficient infinite-depth graph neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 18762–18773.

[Liu et al., 2022] Liu, J., Xu, X., Gan, W., shoushtari, s., and Kamilov, U. (2022). Online deep equilibrium learning for regularization by denoising. In *Advances in Neural Information Processing Systems*, volume 35, pages 25363–25376.

[Lu et al., 2021] Lu, C., Chen, J., Li, C., Wang, Q., and Zhu, J. (2021). Implicit normalizing flows. In *International Conference on Representation Learning*.

[Luo et al., 2025] Luo, C., Wang, H., Xie, T., Jin, Q., Chen, G., Cui, Z.-X., and Liang, D. (2025). Matrix completion-informed deep unfolded equilibrium models for self-supervised k-space interpolation in MRI. *Medical Physics*.

[Marwah et al., 2023] Marwah, T., Pokle, A., Kolter, J. Z., Lipton, Z., Lu, J., and Risteski, A. (2023). Deep equilibrium based neural operators for steady-state PDEs. In *Advances in Neural Information Processing Systems*, volume 36, pages 15716–15737.

[McCallum et al., 2025] McCallum, S., Arora, K., and Foster, J. (2025). Reversible deep equilibrium models. *arXiv*, https://arxiv.org/abs/2509.12917.

[Mehta et al., 2025] Mehta, A., Kitichotkul, R., Goyal, V. K., and Tachella, J. (2025). Equivariant deep equilibrium models for imaging inverse problems. *arXiv*, https://arxiv.org/abs/2511.18667.

[Micaelli et al., 2023] Micaelli, P., Vahdat, A., Yin, H., Kautz, J., and Molchanov, P. (2023). Recurrence without recurrence: Stable video landmark detection with deep equilibrium models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22814–22825.

[Miyato et al., 2018] Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. In *International Conference on Representation Learning*.

[Nguyen and Mauch, 2022] Nguyen, B. and Mauch, L. (2022). Efficient training of deep equilibrium models. In *Hardware Aware Efficient Training ICML workshop*.

[Ni et al., 2023] Ni, J., Bai, Y., Zhang, W., Yao, T., and Mei, T. (2023). Deep equilibrium multimodal fusion. *arXiv*, https://arxiv.org/abs/2306.16645.

[Niculae et al., 2018] Niculae, V., Martins, A., Blondel, M., and Cardie, C. (2018). SparseMAP: differentiable sparse structured inference. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 3799–3808.

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

[Pacheco and Camponogara, 2024] Pacheco, B. M. and Camponogara, E. (2024). Solving differential equations using physics-informed deep equilibrium models. In *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, pages 1883–1888.

[Pal et al., 2023] Pal, A., Edelman, A., and Rackauckas, C. (2023). Continuous deep equilibrium models: Training neural ODEs faster by integrating them to infinity. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–9.

[Park et al., 2025] Park, C. Y., Gan, W., Zou, Z., Hu, Y., Sun, Z., and Kamilov, U. S. (2025). Efficient model-based deep learning via network pruning and fine-tuning. *Journal of Mathematical Imaging and Vision*, 67(2).

[Park et al., 2022] Park, J., Choo, J., and Park, J. (2022). Convergent graph solvers. In *International Conference on Representation Learning*.

[Paszke et al., 2019] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: an imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pages 8024–8035.

[Peaceman and Rachford, 1955] Peaceman, D. W. and Rachford, Jr., H. H. (1955). The numerical solution of parabolic and elliptic differential equations. *Journal of the Society for Industrial and Applied Mathematics*, 3(1):28–41.

[Pineda, 1987] Pineda, F. J. (1987). Generalization of back propagation to recurrent and higher order neural networks. In *Advances in Neural Information Processing Systems*, volume 0, pages 602–611.

[Pokle et al., 2022] Pokle, A., Geng, Z., and Kolter, J. Z. (2022). Deep equilibrium approaches to diffusion models. In *Advances in Neural Information Processing Systems*, volume 35, pages 37975–37990.

[Pramanik and Jacob, 2022] Pramanik, A. and Jacob, M. (2022). Improved model based deep learning using monotone operator learning (MOL). In *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*, pages 1–4.

[Qiao et al., 2020] Qiao, Y., Liang, J., Koltun, V., and Lin, M. C. (2020). Scalable differentiable physics for learning and control. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 7847–7856.

[Qin et al., 2023] Qin, Z., Yang, S., and Zhong, Y. (2023). Hierarchically gated recurrent neural network for sequence modeling. In *Advances in Neural Information Processing Systems*, volume 36, pages 33202–33221.

[Ramzi et al., 2022] Ramzi, Z., Mannel, F., Bai, S., Starck, J., Ciuciu, P., and Moreau, T. (2022). SHINE: sharing the inverse estimate from the forward pass for bi-level optimization and implicit models. In *International Conference on Representation Learning*.

[Revay et al., 2020] Revay, M., Wang, R., and Manchester, I. R. (2020). Lipschitz bounded equilibrium networks. *arXiv*, https://arxiv.org/abs/2010.01732.

[Schleich et al., 2024] Schleich, P., Skreta, M., Kristensen, L. B., Vargas-Hernández, R. A., and Aspuru-Guzik, A. (2024). Quantum deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 37, pages 31940–31967.

[Shenoy et al., 2025] Shenoy, V. R., Lohit, S., Mansour, H., Chellappa, R., and Marks, T. K. (2025). Recovering pulse waves from video using deep unrolling and deep equilibrium models. *arXiv*, https://arxiv.org/abs/2503.17269.

[Shoushtari et al., 2022] Shoushtari, S., Liu, J., Hu, Y., and Kamilov, U. S. (2022). Deep model-based architectures for inverse problems under mismatched priors. *IEEE Journal on Selected Areas in Information Theory*, 3(3):468–480.

[Sittoni and Tudisco, 2024] Sittoni, P. and Tudisco, F. (2024). Subhomogeneous deep equilibrium models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 45794–45812.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

[Srivastava et al., 2015a] Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015a). Highway networks. In *ICML 2015 Deep Learning workshop*, volume https://arxiv.org/abs/1505.00387.

[Srivastava et al., 2015b] Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015b). Training very deep networks. In *Advances in Neural Information Processing Systems*, volume 28, pages 2377–2385.

[Tian and Lian, 2024] Tian, H. and Lian, L. (2024). GSURE-based unsupervised deep equilibrium model learning for large-scale channel estimation. In *GLOBECOM 2024 - 2024 IEEE Global Communications Conference*, pages 4938–4943.

[Tian and Lian, 2025] Tian, H. and Lian, L. (2025). Unsupervised deep equilibrium model learning for large-scale channel estimation with performance guarantees. *arXiv*, https://arxiv.org/abs/2508.10546.

[Truong, 2025] Truong, L. V. (2025). Global convergence rate of deep equilibrium models with general activations. *Transactions on Machine Learning Research*.

[Tschiatschek et al., 2018] Tschiatschek, S., Sahin, A., and Krause, A. (2018). Differentiable submodular maximization. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 2731–2738.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008.

[Wang et al., 2019] Wang, P., Donti, P. L., Wilder, B., and Kolter, J. Z. (2019). SATNet: bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 6545–6554.

[Wang et al., 2023] Wang, S., Teng, Y., and Wang, L. (2023). Deep equilibrium object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6296–6306.

[Wang et al., 2020] Wang, T., Zhang, X., and Sun, J. (2020). Implicit feature pyramid network for object detection. *arXiv*, https://arxiv.org/abs/2012.13563.

[Wang et al., 2024a] Wang, Z., Cheng, L., Moure, P., Hahn, N., and Liu, S.-C. (2024a). DeltaDEQ: exploiting heterogeneous convergence for accelerating deep equilibrium iterations. In *Advances in Neural Information Processing Systems*, volume 37, pages 57564–57590.

[Wang et al., 2024b] Wang, Z., Liu, C., Zou, N., Zhang, H., Wei, X., Huang, L., Wu, L., and Shao, B. (2024b). Infusing self-consistency into density functional theory hamiltonian prediction via deep equilibrium models. In *Advances in Neural Information Processing Systems*, pages 89652–89681.

[Wei and Kolter, 2022] Wei, C. and Kolter, J. Z. (2022). Certified robustness for deep equilibrium models via interval bound propagation. In *International Conference on Representation Learning*.

[Winston and Kolter, 2020] Winston, E. and Kolter, J. Z. (2020). Monotone operator equilibrium networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 10718–10728.

[Wu and He, 2018] Wu, Y. and He, K. (2018). Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19.

[Yang et al., 2023] Yang, Z., Li, P., Pang, T., and Liu, Y. (2023). Improving adversarial robustness of deep equilibrium models with explicit regulations along the neural dynamics. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 39349–39364.

[Yang et al., 2022] Yang, Z., Pang, T., and Liu, Y. (2022). A closer look at the adversarial robustness of deep equilibrium models. In 35, editor, *Advances in Neural Information Processing Systems*, pages 10448–10461.

[Yu and Dansereau, 2024] Yu, Y. and Dansereau, R. M. (2024). MsDC-DEQ-Net: deep equilibrium model (DEQ) with multiscale dilated convolution for image compressive sensing (CS). *IET Signal Processing*, 2024(1):6666549.

[Yuan et al., 2024] Yuan, D., Wu, S., Tang, H., Yang, L., and Peng, C. (2024). A peaceman-rachford splitting approach with deep equilibrium network for channel estimation. In *2024 IEEE 24th International Conference on Communication Technology (ICCT)*, pages 1525–1533.

[Zhang et al., 2025] Zhang, Y., Zhao, Y., and Tao, Q. (2025). Bridging classical and learning-based iterative registration through deep equilibrium models. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2025*.

[Zhao et al., 2021] Zhao, X., Zhang, Z., Zhang, Z., Wu, L., Jin, J., Zhou, Y., Jin, R., Dou, D., and Yan, D. (2021). Expressive 1-lipschitz neural networks for robust multiple graph learning against adversarial attacks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 12719–12735.

[Zhao et al., 2023] Zhao, Y., Zheng, S., and Yuan, X. (2023). Deep equilibrium models for video snapshot compressive imaging. In *Proceedings of the AAAI Conference on Artificial Intelligence,*, volume 37, pages 3642–3650.

[Zou et al., 2023] Zou, Z., Liu, J., Wohlberg, B., and Kamilov, U. S. (2023). Deep equilibrium learning of explicit regularizers for imaging inverse problems. *IEEE Open Journal of Signal Processing*, 4:390–398.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes] See Section 3.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Not Applicable]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes] The code is available at https://github.com/iiduka-researches/Lipschitz_mdeq.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Yes] See Appendix B.
   (c) Clear explanations of any assumptions. [Not Applicable]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes] See Appendix C.1
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes] See Appendix C.1.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes] See Section 4.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes] See Appendix C.1.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes] See Appendix C.1.
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes] The code is available at `https://github.com/iiduka-researches/Lipschitz_mdeq`.
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Lipschitz Multiscale Deep Equilibrium models: A Theoretically Guaranteed and Accelerated Approach Supplementary Materials

A OTHER RELATED WORKS

Other Theory Other research on theoretical aspects of DEQs includes work related to neural tangent kernel theory [Feng and Kolter, 2023, Agarwala and Schoenholz, 2022, Gao et al., 2022, Ling et al., 2023, Truong, 2025, Ling et al., 2024]. Additionally, Kawaguchi proved that under certain assumptions, gradient descent converges to a global optimum for minimizing empirical loss in linear DEQs [Kawaguchi, 2021]. Also, the gradient descent dynamics of DEQ has been studied in the simple settings of linear and single-index models [Dandapanthula and Ramdas, 2025]. Gao et al. explored the infinite-width behavior of deep equilibrium neural networks, and found a limiting Gaussian process behavior that occurs even when depth and width limits are interchanged [Gao et al., 2023].

Application The defining feature of a DEQ is its ability to achieve both high performance and low memory consumption, making it applicable to a remarkably diverse range of tasks. Particularly notable is its application to inverse problems such as wireless channel estimation [Yuan et al., 2024, Tian and Lian, 2024, Tian and Lian, 2025], MRI and general imaging reconstruction [Gilton et al., 2021, Liu et al., 2022, Pramanik and Jacob, 2022, Zou et al., 2023, Park et al., 2025, Luo et al., 2025], CT reconstruction [Bubba et al., 2025], self-supervised imaging reconstruction [Mehta et al., 2025], MPI reconstruction [Güngör et al., 2024], video snapshot compressive imaging [Zhao et al., 2023], Poisson inverse problems [Daniele et al., 2025], Plug-and-Play image reconstruction [Chandler et al.,], medical image registration [HU et al., 2024, Zhang et al., 2025], imaging photoplethysmography (iPPG) [Shenoy et al., 2025], inverse problems under model mismatch [Shoushtari et al., 2022, Hu et al., 2023, Guan et al., 2024], and music source separation [Koyama et al., 2022]. Other applications include diffusion models [Pokle et al., 2022, Geng et al., 2023, Bai and Melas-Kyriazi, 2024], image restoration [Chen et al., 2023, Cao et al., 2024], object detection [Wang et al., 2020, Wang et al., 2023], landmark detection [Micaelli et al., 2023], optical flow estimation [Bai et al., 2022a], point cloud classification and point cloud completion [Geuter et al., 2025], learning solution operators for steady-state PDEs [Marwah et al., 2023], parameterized quantum circuits [Schleich et al., 2024], quantum Hamiltonian prediction [Wang et al., 2024b], molecular dynamics simulations [Burger et al., 2025], initial value problems of ordinary differential equations [Pacheco and Camponogara, 2024], image compressive sensing [Yu and Dansereau, 2024], energy minimization in Hopfield networks [Goemaere et al., 2024], multimodal fusion [Ni et al., 2023], federated learning [Gkillas et al., 2023], and video semantic segmentation and video object detection [Ertenl et al., 2022]. Geng et al. have provided a convenient PyTorch implementation of DEQ [Geng and Kolter, 2023].

Adversarial Robustness Many previous studies focused on the adversarial robustness of DEQ. Chen et al. examined the robustness of monDEQ using its semi-algebraic representation [Chen et al., 2021]. Pabbaraju et al. estimated the Lipschitz constant of monDEQ and demonstrated its robustness against adversarial attacks [Pabbaraju et al., 2021]. Jafarpour et al. studied the stability of fixed-point convergence in a DEQ theoretically using non-Euclidean contraction theory [Jafarpour et al., 2021]. Wei and Kolter proposed a new DEQ called IBP-monDEQ that preserves the existence and uniqueness of fixed points for monDEQ while also providing theoretical guarantees for robustness from the perspective of interval bound propagation [Wei and Kolter, 2022]. Li et al. proposed a certifiable DEQ called CerDEQ [Li et al., 2022]. Yang et al. examined correct attack methods to avoid misjudging the adversarial robustness of DEQs [Yang et al., 2022]. Havens et al. verified the $\ell 2$-certified robustness of DEQs from the perspective of Lipschitz boundaries [Havens et al., 2023]. Jafarpour et al. evaluated the robustness of implicit models using mixed monotone systems theory and contraction theory [Jafarpour et al., 2022]. Yang et al. improved the robustness of DEQs from the perspective of neural dynamics [Yang et al., 2023]. Chu et al. proposed LyaDEQ, a robust DEQ, using Lyapunov functions [Chu et al., 2024]. Gao et al. generalized

provable robustness to large-scale datasets [Gao et al., 2024].

Implicit Layers Family DEQs are positioned within the broader family of implicit layers. Implicit layers is a class of models whose outputs are defined implicitly as the solution to an equation or optimization problem, rather than being computed through a fixed sequence of operations. The history of implicit layers in deep learning dates back to the late 1980s, originating from a paper by Pineda and Almeida [Pineda, 1987, Almeida, 1987], where it is referred to as recurrent back-propagation [Liao et al., 2018]. The family includes Neural ODEs (solutions to differential equations) [Chen et al., 2018, Grathwohl et al., 2019, Dupont et al., 2019], differentiable physics engines [de Avila Belbute-Peres et al., 2018, Qiao et al., 2020], and differentiable optimization layers (solutions to optimization problems) such as OptNet [Amos and Kolter, 2017], differentiable submodular models [Djolonga and Krause, 2017, Tschiatschek et al., 2018], SparseMAP [Niculae et al., 2018], SATNet [Wang et al., 2019], and differentiable convex optimization [Agrawal et al., 2019]. More generally, the deep declarative networks framework [Gould et al., 2022] provides a unifying theory that encompasses such differentiable optimization layers, defining network outputs as solutions to optimization problems. Furthermore, around the same time as DEQs, Ghaoui et al. proposed an implicit deep learning framework defining outputs via fixed-point equations and demonstrated its advantages in robustness and interpretability [Ghaoui et al., 2021]. These models often exhibit compatible functionalities. For example, Ding et al. demonstrated that DEQs and NeuralODE are two sides of the same coin [Ding et al., 2023], while Pal et al. proposed a new architecture that redefines DEQ as an infinite-time neural ODE to reduce training costs [Pal et al., 2023]. In addition, there are several applications, such as graph neural networks [Gu et al., 2020, Liu et al., 2021, Park et al., 2022, Chen et al., 2022] and generative models [Lu et al., 2021].

Lipschitz Networks Our approach is of the kind that controls the model's Lipschitz constant, and there have been several prior studies for explicit models [Zhao et al., 2021, Béthune et al., 2022]. Existing approaches for enforcing Lipschitz constraints fall into two categories: regularization constraints and architecture constraints. Regularization approaches [Drucker and Le Cun, 1992, Gulrajani et al., 2017] exhibit good practical performance but do not reliably enforce Lipschitz constraints. Architecture-based approaches impose restrictions on the operator norm of weights [Cisse et al., 2017, Miyato et al., 2018]. These reliably satisfy the Lipschitz constraint but often come at the cost of reduced expressiveness. Anil et al. proposed a new activation function intended to make neural networks into Lipschitz functions without sacrificing expressiveness [Anil et al., 2019]. In the context of DEQs, Jacobian regularization [Bai et al., 2021] corresponds to a regularization approach, while our study's approach is architecture-based.

B DERIVATION OF LIPSCHITZ CONSTANT FOR EACH OPERATION

B.1 Normalization

Lemma B.1. *Let GN*: $\mathbb{R}^d \to \mathbb{R}^d$ *be the group normalization operation, i.e.,*

$$GN(\boldsymbol{z})_i = \gamma \frac{z_i - \mu_{\boldsymbol{z}}}{\sqrt{\sigma_{\boldsymbol{z}}^2 + \epsilon}} + \beta,$$

where $\mu_{\boldsymbol{z}} := \frac{1}{d}\sum_{i\in[d]} z_i$ *and* $\sigma_{\boldsymbol{z}}^2 := \frac{1}{d}\sum_{i\in[d]}(z_i - \mu_{\boldsymbol{z}})^2$ *are the mean and variance, and* $\gamma, \beta \in \mathbb{R}$ *are learnable affine parameters. Then, GN is a* $\frac{|\gamma|}{\sqrt{\epsilon}}$*-Lipschitz function.*

Proof. If the mapping $f\colon \mathbb{R}^d \to \mathbb{R}^d$ is differentiable, then the Lipschitz constant L_f of f is characterized by the operator norm of the Jacobian matrix J, and $L_f = \sup_{\boldsymbol{x}\in\mathbb{R}^d} \|J(\boldsymbol{x})\|_2$ holds. Therefore, we first show that the operation of GN is differentiable.

All elementary operations used to define GN are differentiable on their domains and remain well-defined because $\epsilon > 0$. Concretely, the following are true:

- The maps $\boldsymbol{z} \mapsto \mu_{\boldsymbol{z}} = \frac{1}{d}\mathbf{1}^\top \boldsymbol{z}$ and $\boldsymbol{z} \mapsto \boldsymbol{z} - \mu_{\boldsymbol{z}}\mathbf{1}$ are linear, hence differentiable to every order.
- The map $\boldsymbol{z} - \mu_{\boldsymbol{z}}\mathbf{1} \mapsto (\boldsymbol{z} - \mu_{\boldsymbol{z}}\mathbf{1})^\top(\boldsymbol{z} - \mu_{\boldsymbol{z}}\mathbf{1})$ is a polynomial, hence differentiable to every order.

- Since $\sigma_{\boldsymbol{z}}^2 = \frac{1}{d}(\boldsymbol{z} - \mu_{\boldsymbol{z}}\mathbf{1})^\top(\boldsymbol{z} - \mu_{\boldsymbol{z}}\mathbf{1}) \geq 0$ and $\epsilon > 0$, the scalar function $t \mapsto \sqrt{t+\epsilon}$ is smooth on the interval $[0, \infty)$; in particular, it admits derivatives of all orders when evaluated at $t = \sigma_{\boldsymbol{z}}^2$.

Note that $\mathbf{1}$ denotes a vector whose elements are all 1. Therefore, GN is differentiable at every $\boldsymbol{z} \in \mathbb{R}^d$, and in fact derivatives of arbitrary order exist (because each constituent map admits derivatives of arbitrary order and derivatives are closed under composition). Moreover, the denominator

$$s(\boldsymbol{z}) := \sqrt{\sigma_{\boldsymbol{z}}^2 + \epsilon}$$

satisfies $s(\boldsymbol{z}) \geq \sqrt{\epsilon} > 0$ for all $\boldsymbol{z}$, so no division by zero occurs; consequently, every partial derivative of the Jacobian entries is continuous and the Jacobian matrix $J(\boldsymbol{z})$ depends continuously on $\boldsymbol{z}$.

Next, we derive the Jacobian of GN. For simplicity, we introduce $\hat{z}_i := \frac{z_i - \mu_{\boldsymbol{z}}}{\sqrt{\sigma_{\boldsymbol{z}}^2+\epsilon}}$ and obtain $\mathrm{GN}(\boldsymbol{z})_i := \gamma\hat{z}_i + \beta$. As preparation, we will calculate several derivatives. From the definition of $\mu_{\boldsymbol{z}}$, we have

$$\frac{\partial \mu_{\boldsymbol{z}}}{\partial z_k} = \frac{\partial}{\partial z_k}\left(\frac{1}{d}\sum_{i\in[d]} z_i\right) = \frac{1}{d}, \text{ and } \frac{\partial \mu_{\boldsymbol{z}}^2}{\partial z_k} = \frac{\partial \mu_{\boldsymbol{z}}^2}{\partial \mu_{\boldsymbol{z}}} \cdot \frac{\partial \mu_{\boldsymbol{z}}}{\partial z_k} = \frac{2\mu_{\boldsymbol{z}}}{d}.$$

In addition, from the definition of $\sigma_{\boldsymbol{z}}^2$, we have

$$\frac{\partial \sigma_{\boldsymbol{z}}^2}{\partial z_k} = \frac{\partial}{\partial z_k}\left(\frac{1}{d}\sum_{i\in[d]} z_i^2 - \mu_{\boldsymbol{z}}^2\right) = \frac{\partial}{\partial z_k}\left(\frac{1}{d}\sum_{i\in[d]} z_i^2\right) - \frac{\partial}{\partial z_k}\mu_{\boldsymbol{z}}^2 = \frac{2}{d}(z_k - \mu_{\boldsymbol{z}}).$$

Moreover, from the definition of $s(\boldsymbol{z})$,

$$\frac{\partial s(\boldsymbol{z})}{\partial z_k} = \frac{\partial s(\boldsymbol{z})}{\partial \sigma_{\boldsymbol{z}}^2} \cdot \frac{\partial \sigma_{\boldsymbol{z}}^2}{\partial z_k} = \frac{(\sigma_{\boldsymbol{z}}^2 + \epsilon)^{-\frac{1}{2}}}{2} \cdot \frac{2}{d}(z_k - \mu_{\boldsymbol{z}}) = \frac{\hat{z}_k}{d}.$$

Finally, from the quotient rule,

$$\begin{aligned}\frac{\partial \hat{z}_i}{\partial z_k} &= \frac{\partial}{\partial z_k}\frac{z_i - \mu_{\boldsymbol{z}}}{s(\boldsymbol{z})} = \frac{\frac{\partial}{\partial z_k}(z_i - \mu_{\boldsymbol{z}})s(\boldsymbol{z}) - (z_i - \mu_{\boldsymbol{z}})\frac{\partial}{\partial z_k}s(\boldsymbol{z})}{s(\boldsymbol{z})^2} = \frac{\left(\frac{\partial}{\partial z_k}z_i - \frac{\partial}{\partial z_k}\mu_{\boldsymbol{z}}\right)s(\boldsymbol{z}) - (z_i - \mu_{\boldsymbol{z}})\frac{\hat{z}_k}{d}}{s(\boldsymbol{z})^2} \\ &= \frac{\left(\delta_{ik} - \frac{1}{d}\right)s(\boldsymbol{z}) - (z_i - \mu_{\boldsymbol{z}})\frac{\hat{z}_k}{d}}{s(\boldsymbol{z})^2} = \frac{1}{s(\boldsymbol{z})}\left(\delta_{ik} - \frac{1}{d}\right) - \frac{1}{s(\boldsymbol{z})}\frac{z_i - \mu_{\boldsymbol{z}}}{s(\boldsymbol{z})}\frac{\hat{z}_k}{d} = \frac{1}{s(\boldsymbol{z})}\left\{\left(\delta_{ik} - \frac{1}{d}\right) - \frac{1}{d}\hat{z}_i\hat{z}_k\right\},\end{aligned}$$

where δ_{ik} is the Kronecker delta. Then, from the chain rule, we have

$$J_{ik} = \frac{\partial \mathrm{GN}(\boldsymbol{z})_i}{\partial z_k} = \frac{\partial \mathrm{GN}(\boldsymbol{z})_i}{\partial \hat{z}_i} \cdot \frac{\partial \hat{z}_i}{\partial z_k} = \gamma\frac{\partial \hat{z}_i}{\partial z_k} = \frac{\gamma}{s(\boldsymbol{z})}\left\{\left(\delta_{ik} - \frac{1}{d}\right) - \frac{1}{d}\hat{z}_i\hat{z}_k\right\}.$$

Therefore,

$$J(\boldsymbol{z}) = \frac{1}{s(\boldsymbol{z})}\operatorname{diag}(\gamma)\left\{\left(I - \frac{1}{d}\mathbf{1}\mathbf{1}^\top\right) - \frac{1}{d}\hat{\boldsymbol{z}}\hat{\boldsymbol{z}}^\top\right\}.$$

From submultiplicativity, we have

$$\|J(\boldsymbol{z})\|_2 \leq \left\|\frac{1}{s(\boldsymbol{z})}\operatorname{diag}(\gamma)\right\|_2 \left\|\left(I - \frac{1}{d}\mathbf{1}\mathbf{1}^\top\right) - \frac{1}{d}\hat{\boldsymbol{z}}\hat{\boldsymbol{z}}^\top\right\|_2 \leq \frac{|\gamma|}{s(\boldsymbol{z})}\Bigg\|\underbrace{\left(I - \frac{1}{d}\mathbf{1}\mathbf{1}^\top\right) - \frac{1}{d}\hat{\boldsymbol{z}}\hat{\boldsymbol{z}}^\top}_{=:A(\boldsymbol{z})}\Bigg\|_2.$$

Since $\|A(\boldsymbol{z})\|_2$ is the largest eigenvalue of $A(\boldsymbol{z})$, let us consider the eigenvalues of $A(\boldsymbol{z})$.

(i) Multiplying $A(\boldsymbol{z})$ by $\hat{\boldsymbol{z}}$ gives

$$A(\boldsymbol{z})\hat{\boldsymbol{z}} = \left(I - \frac{1}{d}\mathbf{1}\mathbf{1}^\top\right)\hat{\boldsymbol{z}} - \frac{1}{d}\hat{\boldsymbol{z}}\hat{\boldsymbol{z}}^\top\hat{\boldsymbol{z}} = \left(\hat{\boldsymbol{z}} - \frac{1}{d}\mathbf{1}\left(\mathbf{1}^\top\hat{\boldsymbol{z}}\right)\right) - \frac{1}{d}\hat{\boldsymbol{z}}\|\hat{\boldsymbol{z}}\|_2^2 = \left(1 - \frac{\|\hat{\boldsymbol{z}}\|_2^2}{d}\right)\hat{\boldsymbol{z}},$$

where we use $\mathbf{1}^\top \hat{\boldsymbol{z}} = 0$. Here, from the definition of $\hat{\boldsymbol{z}}$, we have

$$\|\hat{\boldsymbol{z}}\|_2^2 = \left\| \frac{\boldsymbol{z} - \mu_{\boldsymbol{z}} \mathbf{1}}{\sqrt{\sigma^2 + \epsilon}} \right\|_2^2 = \frac{1}{\sigma_{\boldsymbol{z}}^2 + \epsilon} \|\boldsymbol{z} - \mu_{\boldsymbol{z}} \mathbf{1}\|_2^2 = \frac{1}{\sigma_{\boldsymbol{z}}^2 + \epsilon} \sum_{i \in [d]} (z_i - \mu_{\boldsymbol{z}})^2 = \frac{d\sigma_{\boldsymbol{z}}^2}{\sigma_{\boldsymbol{z}}^2 + \epsilon}.$$

Hence,

$$A(\boldsymbol{z})\hat{\boldsymbol{z}} = \left(1 - \frac{\sigma_{\boldsymbol{z}}^2}{\sigma_{\boldsymbol{z}}^2 + \epsilon}\right) \hat{\boldsymbol{z}} = \frac{\epsilon}{\sigma_{\boldsymbol{z}}^2 + \epsilon} \hat{\boldsymbol{z}}.$$

Therefore, $\hat{\boldsymbol{z}}$ is an eigenvector, and its eigenvalue is $\frac{\epsilon}{\sigma_{\boldsymbol{z}}^2 + \epsilon} \in (0, 1]$.

(ii) Multiplying $A(\boldsymbol{z})$ by $\mathbf{1}$ gives

$$A(\boldsymbol{z})\mathbf{1} = \left(I - \frac{1}{d}\mathbf{1}\mathbf{1}^\top\right)\mathbf{1} - \frac{1}{d}\hat{\boldsymbol{z}}\hat{\boldsymbol{z}}^\top \mathbf{1} = \mathbf{1} - \frac{1}{d}\mathbf{1}\left(\mathbf{1}^\top \mathbf{1}\right) - \frac{1}{d}\hat{\boldsymbol{z}}\left(\hat{\boldsymbol{z}}^\top \mathbf{1}\right) = \mathbf{1} - \frac{1}{d}\mathbf{1} \cdot d - \frac{1}{d}\hat{\boldsymbol{z}} \cdot 0 = \mathbf{0},$$

where we use $\mathbf{1}^\top \hat{\boldsymbol{z}} = 0$. Therefore, $\mathbf{1}$ is an eigenvector, and its eigenvalue is 0.

(iii) Multiplying $A(\boldsymbol{z})$ by a vector $\boldsymbol{v}$ that is perpendicular to both $\hat{\boldsymbol{z}}$ and $\mathbf{1}$,

$$A(\boldsymbol{z})\boldsymbol{v} = \left(I - \frac{1}{d}\mathbf{1}\mathbf{1}^\top\right)\boldsymbol{v} - \frac{1}{d}\hat{\boldsymbol{z}}\hat{\boldsymbol{z}}^\top \boldsymbol{v} = \boldsymbol{v} - \frac{1}{d}\mathbf{1}\left(\mathbf{1}^\top \boldsymbol{v}\right) - \frac{1}{d}\hat{\boldsymbol{z}}\left(\hat{\boldsymbol{z}}^\top \boldsymbol{v}\right) = \boldsymbol{v} - \frac{1}{d}\mathbf{1} \cdot 0 - \frac{1}{d}\hat{\boldsymbol{z}} \cdot 0 = \boldsymbol{v},$$

where we use $\mathbf{1}^\top \boldsymbol{v} = 0$ and $\hat{\boldsymbol{z}}^\top \boldsymbol{v} = 0$. Therefore, $\boldsymbol{v}$ is an eigenvector, and its eigenvalue is 1.

Hence, the largest eigenvalue of $A(\boldsymbol{z})$ is 1, i.e., $\|A(\boldsymbol{z})\|_2 = 1$. Consequently, we have

$$\|J(\boldsymbol{z})\|_2 \leq \frac{|\gamma|}{s(\boldsymbol{z})} = \frac{|\gamma|}{\sqrt{\sigma_{\boldsymbol{z}}^2 + \epsilon}} \leq \frac{|\gamma|}{\sqrt{\epsilon}}.$$

This completes the proof. □

Lemma B.2. *Let MGN*: $\mathbb{R}^d \to \mathbb{R}^d$ *be the mean-only group normalization operation, i.e.,*

$$MGN(\boldsymbol{z})_i = \gamma(z_i - \mu_{\boldsymbol{z}}) + \beta,$$

where $\mu_{\boldsymbol{z}} := \frac{1}{d}\sum_{i \in [d]} z_i$ *is the mean, and* $\gamma, \beta \in \mathbb{R}$ *are learnable affine parameters. Then, MGN is a* $|\gamma|$*-Lipschitz function.*

Proof. By the same argument as Lemma B.1, since MGN is differentiable, it suffices to consider its Jacobian matrix:

$$J_{ik} = \frac{\partial \mathrm{MGN}(\boldsymbol{z})_i}{\partial z_k} = \frac{\partial}{\partial z_k}\left\{\gamma(z_i - \mu_{\boldsymbol{z}}) + \beta\right\} = \gamma\delta_{ik} - \frac{1}{d},$$

where δ_{ik} is the Kronecker delta. Therefore,

$$J(\boldsymbol{z}) = \mathrm{diag}(\gamma)\left(I - \frac{1}{d}\mathbf{1}\mathbf{1}^\top\right).$$

From submultiplicativity, we have

$$\|J(\boldsymbol{z})\|_2 \leq \|\mathrm{diag}(\gamma)\|_2 \left\| I - \frac{1}{d}\mathbf{1}\mathbf{1}^\top \right\|_2 = |\gamma| \Big\| \underbrace{I - \frac{1}{d}\mathbf{1}\mathbf{1}^\top}_{=:P} \Big\|_2.$$

Let us consider the eigenvalues of P. Here, from

$$P^\top = \left(I - \frac{1}{d}\mathbf{1}\mathbf{1}^\top\right)^\top = I - \frac{1}{d}\mathbf{1}\mathbf{1}^\top = P,$$

and

$$P^2 = \left(I - \frac{1}{d}\mathbf{1}\mathbf{1}^\top\right)\left(I - \frac{1}{d}\mathbf{1}\mathbf{1}^\top\right) = I - \frac{1}{d}\mathbf{1}\mathbf{1}^\top - \frac{1}{d}\mathbf{1}\mathbf{1}^\top + \frac{1}{d^2}\mathbf{1}\mathbf{1}^\top\mathbf{1}\mathbf{1}^\top = I - \frac{1}{d}\mathbf{1}\mathbf{1}^\top = P,$$

P is a projection matrix, and its eigenvalues are 0 and 1. Therefore, we have $\|P\|_2 = 1$ and

$$\|J(\boldsymbol{z})\|_2 \leq |\gamma|.$$

This completes the proof. □

B.2 Activation Function

Lemma B.3. *Let ReLU*: $\mathbb{R}^d \to \mathbb{R}^d$ *be the ReLU function, i.e., ReLU*$(\boldsymbol{z})_i := \max\{0, z_i\}$. *Then, ReLU is a* 1*-Lipschitz function.*

Proof. As preparation, consider the scalar version of the ReLU function $g\colon \mathbb{R} \to \mathbb{R}$, i.e., $g(z) := \max\{0, z\}$.

- If $z_1 \geq 0$ and $z_2 \geq 0$, then we have
$$|g(z_1) - g(z_2)| = |z_1 - z_2|.$$
- If $z_1 \leq 0$ and $z_2 \leq 0$, then we have
$$|g(z_1) - g(z_2)| = 0 \leq |z_1 - z_2|.$$
- If $z_1 \geq 0$ and $z_2 \leq 0$, then we have
$$|g(z_1) - g(z_2)| = |z_1 - 0| = z_1 \leq z_1 - z_2 = |z_1 - z_2|.$$
- If $z_1 \leq 0$ and $z_2 \geq 0$, then we have
$$|g(z_1) - g(z_2)| = |0 - z_2| = z_2 \leq z_2 - z_1 \leq |z_1 - z_2|.$$

Hence, for all $z_1, z_2 \in \mathbb{R}$,

$$|g(z_1) - g(z_2)| \leq |z_1 - z_2|.$$

From the scalar result, we have for each coordinate

$$|\mathrm{ReLU}(\boldsymbol{x})_i - \mathrm{ReLU}(\boldsymbol{y})_i| \leq |x_i - y_i|.$$

Therefore, we have

$$\|\mathrm{ReLU}(\boldsymbol{x}) - \mathrm{ReLU}(\boldsymbol{y})\|_2 = \left(\sum_{i\in[d]} |\mathrm{ReLU}(\boldsymbol{x})_i - \mathrm{ReLU}(\boldsymbol{y})_i|^2\right)^{\frac{1}{2}} \leq \left(\sum_{i\in[d]} |x_i - y_i|^2\right)^{\frac{1}{2}} = \|\boldsymbol{x} - \boldsymbol{y}\|_2.$$

This completes the proof. □

Lemma B.4. *Let SReLU*: $\mathbb{R}^d \to \mathbb{R}^d$ *be the SReLU function, i.e., SReLU*$(\boldsymbol{z})_i := \max\{0, az_i\}$ $(a \in (0, 1])$. *Then, SReLU is an a-Lipschitz function.*

Proof. By a similar argument to Lemma B.3, we have for each coordinate

$$|\mathrm{SReLU}(\boldsymbol{x})_i - \mathrm{SReLU}(\boldsymbol{y})_i| \leq a|x_i - y_i|.$$

Therefore, we have

$$\|\mathrm{SReLU}(\boldsymbol{x}) - \mathrm{SReLU}(\boldsymbol{y})\|_2 = \left(\sum_{i\in[d]} |\mathrm{SReLU}(\boldsymbol{x})_i - \mathrm{SReLU}(\boldsymbol{y})_i|^2\right)^{\frac{1}{2}} \leq \left(\sum_{i\in[d]} a^2|x_i - y_i|^2\right)^{\frac{1}{2}} = a\|\boldsymbol{x} - \boldsymbol{y}\|_2.$$

This completes the proof. □

B.3 Dropout

Lemma B.5. *Let Drop*: $\mathbb{R}^d \to \mathbb{R}^d$ *be the variational dropout function, i.e., Drop*$(\boldsymbol{z}) := \frac{1}{1-p}\boldsymbol{m} \odot \boldsymbol{z}$*, where* $p \in (0,1)$ *and* $\boldsymbol{m} \sim$ *Bernouli*(p)*. Then Drop is a* $\frac{1}{1-p}$*-Lipschitz function.*

Proof. Fix a realization of the mask $\boldsymbol{m} = (m_1, \ldots, m_d)^\top$ with $m_i \in \{0,1\}$. For each coordinate $i \in [d]$, we have

$$|\mathrm{Drop}(\boldsymbol{x})_i - \mathrm{Drop}(\boldsymbol{y})_i| = \frac{1}{1-p}|m_i(x_i - y_i)| \leq \frac{1}{1-p}|x_i - y_i|,$$

where we use $|m_i| \leq 1$. Therefore, we have

$$\|\mathrm{Drop}(\boldsymbol{x}) - \mathrm{Drop}(\boldsymbol{y})\|_2 = \left(\sum_{i\in[d]} |\mathrm{Drop}(\boldsymbol{x})_i - \mathrm{Drop}(\boldsymbol{y})_i|^2\right)^{\frac{1}{2}} \leq \left(\sum_{i\in[d]} \left(\frac{1}{1-p}\right)^2 |x_i - y_i|^2\right)^{\frac{1}{2}} = \frac{1}{1-p}\|\boldsymbol{x} - \boldsymbol{y}\|_2.$$

This completes the proof. □

B.4 Convolution

Lemma B.6. *Let Conv*: $\mathbb{R}^d \to \mathbb{R}^d$ *be the convolution function, i.e., Conv*$(\boldsymbol{z}) := W\boldsymbol{z} + \boldsymbol{b}$*, where* W *is a parameter matrix from the convolutional kernel and* $\boldsymbol{b} \in \mathbb{R}^d$ *is a bias term. Then, Conv is a* $\|W\|_2$*-Lipschitz function.*

Proof. From the definition of the operation Conv, for all $\boldsymbol{z}_1, \boldsymbol{z}_2 \in \mathbb{R}^d$, we have

$$\begin{aligned}\|\mathrm{Conv}(\boldsymbol{z}_1) - \mathrm{Conv}(\boldsymbol{z}_2)\|_2 &= \|(W\boldsymbol{z}_1 + \boldsymbol{b}) - (W\boldsymbol{z}_2 + \boldsymbol{b})\|_2 \\ &= \|W(\boldsymbol{z}_1 - \boldsymbol{z}_2)\|_2 \\ &\leq \|W\|_2\|\boldsymbol{z}_1 - \boldsymbol{z}_2\|_2\end{aligned}$$

This completes the proof. □

B.5 Upsample

Lemma B.7. *Let Up*: $\mathbb{R}^{H\times W} \to \mathbb{R}^{sH\times tW}$ *be the upsample operation, where* $s, t \in \mathbb{N}$ *are the vertical and horizontal scale factors, respectively. Then, Up is a* $\sqrt{st}$*-Lipschitz function.*

Proof. When input $\boldsymbol{x} \in \mathbb{R}^{H\times W}$ is upscaled using nearest-neighbor interpolation to construct $\mathrm{Up}(\boldsymbol{x}) \in \mathbb{R}^{sH\times tW}$, each element of $\boldsymbol{x}$ appears $s \times t$ times in $\mathrm{Up}(\boldsymbol{x})$. Therefore, we have

$$\|\mathrm{Up}(\boldsymbol{x}) - \mathrm{Up}(\boldsymbol{y})\|_2 = \sqrt{st \sum_{i\in[H\times W]} (x_i - y_i)^2} = \sqrt{st}\|\boldsymbol{x} - \boldsymbol{y}\|_2.$$

This completes the proof. □

B.6 Backward Pass

Lemma B.8. *Let* $T\colon \mathcal{Z} \to \mathcal{Z}$ *be the mapping appearing in the backward pass, i.e.,* $T(\mathbf{x}) := \mathbf{x}J_{f_{\boldsymbol{\theta}}}(\boldsymbol{z}^\star) + \frac{\partial l}{\partial \boldsymbol{z}^\star}$*. Then,* T *is a* $\|J_{f_{\boldsymbol{\theta}}}(\boldsymbol{z}^\star)\|_2$*-Lipschitz mapping.*

Proof. From the definition of T and $\|\cdot\|_{\mathcal{Z}}$, we have

$$\begin{aligned}\|T(\mathbf{x}) - T(\mathbf{y})\|_{\mathcal{Z}} &= \|(\mathbf{x} - \mathbf{y})J_{f_{\boldsymbol{\theta}}}(\boldsymbol{z}^\star)\|_{\mathcal{Z}} \\ &= \left\|J_{f_{\boldsymbol{\theta}}}(\boldsymbol{z}^\star)^\top(\mathbf{x} - \mathbf{y})^\top\right\|_{\mathcal{Z}} \\ &\leq \left\|J_{f_{\boldsymbol{\theta}}}(\boldsymbol{z}^\star)^\top\right\|_2 \left\|(\mathbf{x} - \mathbf{y})^\top\right\|_{\mathcal{Z}} \\ &\leq \|J_{f_{\boldsymbol{\theta}}}(\boldsymbol{z}^\star)\|_2 \|\mathbf{x} - \mathbf{y}\|_{\mathcal{Z}}.\end{aligned}$$

This completes the proof. □

C ADDITIONAL EXPERIMENTAL RESULTS

C.1 Experimental Settings and Hyperparameters

In our experiments, all models were trained on single NVIDIA GeForce RTX 4090 GPU and Intel Core i9 13900KF CPU. The code is available at https://github.com/iiduka-researches/Lipschitz_mdeq. All CIFAR-10 [Krizhevsky, 2009] experiments in the main text were run for 200 epochs. A batch size of 128 was used except for Jacobian regularization, where a batch size of 96 was employed. Adam [Kingma and Ba, 2015] was used to optimize the empirical loss, with an initial learning rate of 0.001. Other hyperparameters followed prior work, and the configuration is publicly available at https://github.com/iiduka-researches/Lipschitz_mdeq. For all experiments, the Anderson Acceleration fixed-point solver was used for both the forward and backward passes. The hyperparameters for the phantom gradient were set as 5 iterations and a decay rate of 0.5.

C.2 Comparison of Fixed Point Convergence

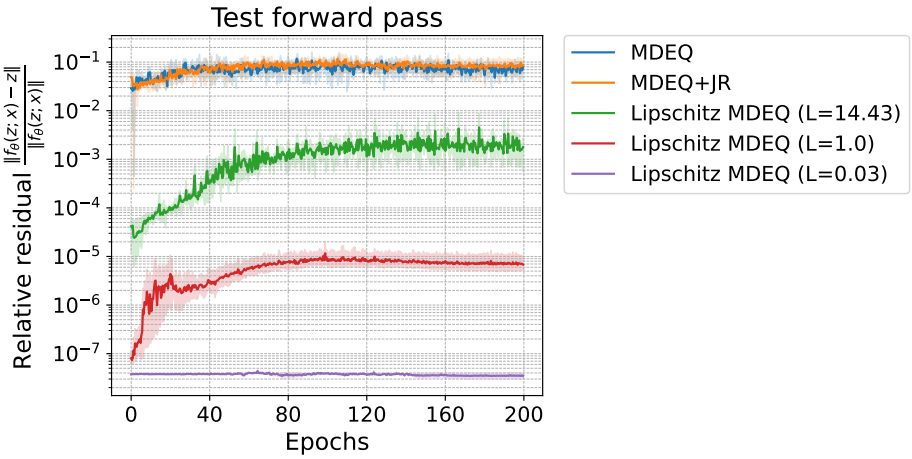


Figure 5: Comparison of fixed-point convergence in the forward pass of testing.

C.3 Experiments on ImageNet

We performed the same experiments on the ImageNet dataset as described in Section 4. In all experiments, we control the Lipschitz constant L by fixing $L_{\text{MGN}} = 1.0$, $L_{\text{Conv}^\star} = 2.0$, $\alpha_1 = 0.5$, $\alpha_2 = 0.3$, $n = 4$, and $p = 0$ and varying $L_{\text{SReLU}} = \{0.1, 0.4\}$. Specifically, $L = \{0.026, 0.794\}$ are obtained when $L_{\text{SReLU}} = \{0.1, 0.4\}$, respectively. All fixed-point iteration evaluation metrics used relative residual, with a threshold of 1e-3. The maximum iteration was set to 14 for both the forward and backward passes.

Table 3: Comparison of different acceleration techniques for MDEQ on ImageNet. We report top-1 and top-5 accuracy, total training speed, and average number of fixed-point iterations (NFEs) and runtime (Time) for the forward and backward passes. Best results in each column are highlighted in **bold**.

Model and method	Size	Speed-up	top-1 Acc.	top-5 Acc.	NFEs (Train Fwd)	NFEs (Train Bwd)	NFEs (Test Fwd)
MDEQ	18M	1.0×	61.83	83.37	14.0	14.0	14.0
Lipschitz MDEQ ($L = 0.794$)	18M	1.15×	62.86	84.13	13.3	12.9	13.3
Lipschitz MDEQ ($L = 0.026$)	18M	1.42×	60.89	82.74	**2.0**	**2.0**	**2.0**
Lipschitz MDEQ ($L = 0.794$) - Conv*	18M	1.34×	**64.21**	**85.01**	7.6	4.4	7.7
Lipschitz MDEQ ($L = 0.026$) - Conv*	18M	**1.48**×	62.95	84.02	**2.0**	**2.0**	**2.0**

In both the forward pass during training and testing and the backward pass during training, Lipschitz MDEQ achieves significantly faster fixed-point convergence than existing methods. However, despite a reduction in the number of fixed-point iterations, the speed-up effect is more modest compared to the case of CIFAR10. This is because we set the maximum number of fixed-point iterations to 14 across all settings to reduce computational cost. Typically, experiments using the ImageNet dataset set the maximum number of iterations to 22–30. Therefore, it is expected that MDEQ's test accuracy could be even higher. Simultaneously, since increasing

the number of iterations does not cause MDEQ's fixed-point iterations to converge, it is anticipated that the acceleration effect of our model will be further emphasized.

The results of the ablation study in Section 4.1 indicate that the setting yielding the greatest experimental acceleration effect is reverting the Lipschitz MDEQ's $\mathrm{Conv}^\star$ operation back to the Conv operation. Therefore, Table 3 also reports the results when the $\mathrm{Conv}^\star$ operation is removed. Even in the case of ImageNet, removing $\mathrm{Conv}^\star$ modifications proved effective, leaving the challenge of better modifications to the convolutional operations.