Stabilizing Equilibrium Models by Jacobian Regularization

Shaojie Bai [1] **Vladlen Koltun** [2] **J. Zico Kolter** [1]

Abstract

Deep equilibrium networks (DEQs) are a new class of models that eschews traditional depth in favor of finding the fixed point of a single non-linear layer. These models have been shown to achieve performance competitive with the state-of-the-art deep networks while using significantly less memory. Yet they are also slower, brittle to architectural choices, and introduce potential instability to the model. In this paper, we propose a regularization scheme for DEQ models that explicitly regularizes the Jacobian of the fixed-point update equations to stabilize the learning of equilibrium models. We show that this regularization adds only minimal computational cost, significantly stabilizes the fixed-point convergence in both forward and backward passes, and scales well to high-dimensional, realistic domains (e.g., WikiText-103 language modeling and ImageNet classification). Using this method, we demonstrate, for the first time, an implicit-depth model that runs with approximately the same speed and level of performance as popular conventional deep networks such as ResNet-101, while still maintaining the constant memory footprint and architectural simplicity of DEQs. Code is available here.

1. Introduction

While conventional deep networks like ResNets (He et al., 2016) and Transformers (Vaswani et al., 2017) rely on hierarchical layer stacking, the recently-proposed deep equilibrium networks (DEQs) (Bai et al., 2019) directly model the "infinite-depth" representation of a single layer f_θ by solving for its fixed point (i.e., "equilibrium") $\mathbf{z}^\star$:

$$\mathbf{z}^\star = f_\theta(\mathbf{z}^\star; \mathbf{x}),$$

where $\mathbf{x}$ is the original input. Importantly, to train these models, one could directly differentiate through the final equilibrium $\mathbf{z}^\star$ by the implicit function theorem (Krantz & Parks, 2012), irrespective of the method used to solve for this equilibrium in the forward pass. Therefore, like other implicit-depth architectures such as Neural ODEs (Chen et al., 2018), DEQs have the notable advantages that their forward passes can rely on any black-box root solvers (e.g., Newton, quasi-Newton, simplest forward iterations), and that their training only consumes $O(1)$ memory. With this formulation, prior works have managed to extend the DEQ framework for multiple large-scale applications, such as language modeling (Bai et al., 2019) and large-scale image classification or segmentation (Bai et al., 2020).

However, these models suffer from a few issues. First, despite their memory efficiency, DEQs are also slower than conventional deep networks that achieve the same level of accuracy. Second, the number of iterations required to solve for the equilibrium quickly grows over the course of training, indicating a trend for approaching instability. Third, the DEQ model is sensitive to architectural choices, and sometimes even small modifications could break the model's stability of convergence. Some recent works have tackled this third issue by exploiting provably convergent layers via monotone operator splitting theories (Winston & Kolter, 2020) and Lipschitz boundedness (Revay et al., 2020). However, these structural solutions rely extensively on specific layer parameterizations, rendering DEQ models unscalable and even more inflexible.

In this paper, we first summarize and provide empirical evidence on all of these downsides of the equilibrium networks that have so far thwarted many from extending DEQs to both broader applications and more architectural variants. To address these issues, we further propose a regularization solution to improve on DEQ models' stability, efficiency and flexibility. Importantly, while prior DEQs adopted regularization methods direcly borrowed from explicit deep networks (e.g., recurrent dropout (Gal & Ghahramani, 2016)), we introduce a simple and theoretically-motivated Jacobian regularization pursuant to DEQ models' implicitness. We will discuss in detail how this Jacobian regularization relates to the contractivity of DEQ's forward non-linear system and backward linear system, and is thus able to effectively stabilize not only forward but also backward dynamics of DEQ networks. There are two immediate benefits of the resulting stability in the dynamics. First, solving a DEQ requires far fewer iterations than before, which makes regularized

[1]Carnegie Mellon University, Pittsburgh PA, USA [2]Intel Labs, USA. Correspondence to: Shaojie Bai <shaojieb@cs.cmu.edu>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

DEQs significantly faster than their unregularized counterparts. Second, this model class becomes much less brittle to architectural variants that would otherwise break the DEQ.

We validate the proposed regularization by experiments on both toy-scale synthetic tasks and large-scale real datasets across domains: word-level language modeling on WikiText-103 (Merity et al., 2017) and high-resolutional image classification on the full ImageNet dataset (Deng et al., 2009). Empirically, our regularized DEQs are generally 2x to 3x faster than prior DEQs, and can be accelerated to be as fast as explicit deep networks (e.g., ResNets, DenseNets, and Transformers). This is the first time that implicit models are accelerated to this level without sacrificing scalability, accuracy, or structural flexibility. With their $O(1)$ memory footprint, this further establishes implicit models as a strong competitor to explicit deep architectures.

2. Background & Related Work

While explicit deep networks hierarchically stack layers to build a computation graph for their forward and backward propagations, implicit models (Amos & Kolter, 2017; Chen et al., 2018; El Ghaoui et al., 2019; Gould et al., 2019; Bai et al., 2019) do not have a prescribed computation graph. Instead these models solve for a non-linear system. One example is the Neural ODE (Chen et al., 2018), which solves an initial-value ODE problem that involves a residual layer. Another example, which is the primary focus of our work, is the deep equilibrium network (DEQ) (Bai et al., 2019), which reduces the forward pass to a root-solving problem. In this section, we introduce the basics of DEQ models and the relevant threads of work, followed by a discussion of prior approaches to regularizing implicit models.

2.1. Deep Equilibrium Models

Given a layer/block f_θ (which may contain a few shallow sublayers) and an input $\mathbf{x}$, a deep equilibrium model aims to approximate an "infinite-level" layer stacking of the form $\mathbf{z}^{[i+1]} = f_\theta(\mathbf{z}^{[i]}; \mathbf{x})$ (where $i = 1, \dots, L$, with $L \to \infty$) by directly solving for its fixed-point representation:

$$\mathbf{z}^\star = f_\theta(\mathbf{z}^\star; \mathbf{x}).$$

One of the appealing properties of this fixed-point formulation is that one can implicitly differentiate through the equilibrium feature, without dependency on any intermediate activations in the forward pass. Formally, given a loss ℓ, one can directly compute the gradient using the final output:

$$\frac{\partial \ell}{\partial (\cdot)} = \frac{\partial \ell}{\partial \mathbf{z}^\star} \left(I - J_{f_\theta}(\mathbf{z}^\star)\right)^{-1} \frac{\partial f_\theta(\mathbf{z}^\star; \mathbf{x})}{\partial (\cdot)},$$

where $J_{f_\theta}(\mathbf{z}^\star)$ is the Jacobian matrix at equilibrium $\mathbf{z}^\star$.

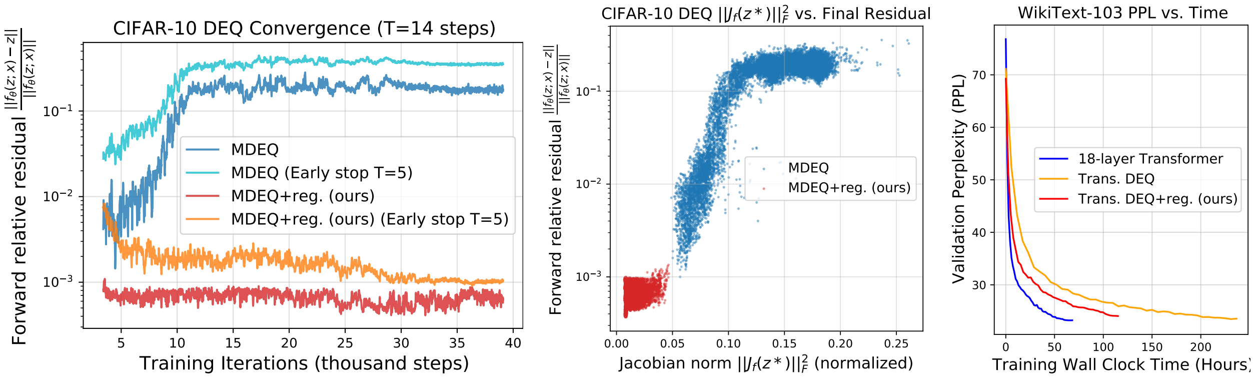
To solve for the equilibrium, Bai et al. (2019) proposed to use Broyden's method (Broyden, 1965) to find the root of $f_\theta(\mathbf{z}^\star; \mathbf{x}) - \mathbf{z}^\star = 0$; later works (Winston & Kolter, 2020; Revay et al., 2020) and a recent tutorial (Duvenaud et al., 2020) have applied other algorithms, such as Peaceman-Rachford splitting (Peaceman & Rachford, 1955) and Anderson acceleration (Anderson, 1965).

Compared to Neural ODEs, deep equilibrium networks have been demonstrated to scale well to large and high-dimensional tasks, such as language modeling, ImageNet classification, and semantic segmentation (Bai et al., 2019; 2020), and are thus more applicable to domains where deep learning has been traditionally successful. However, unlike ODE flows, DEQ networks do not have a unique trajectory, and are not guaranteed to converge. Thus recent works have also begun to examine the stability and other theoretical properties of DEQs. Winston & Kolter (2020) propose a monotone DEQ that has a unique fixed point. Pabbaraju et al. (2021); Revay et al. (2020) further study the Lipschitz boundedness of monotone DEQs. Kawaguchi (2021) analyze the gradient dynamics of a linearized version of DEQs. Lu et al. (2021) apply an invertible equilibrium model to generative modeling via normalizing flows.

2.2. Regularizing Implicit Models

Just like explicit deep networks, implicit networks can overfit to the dataset; but additionally, they can also become unstable. For instance, Neural ODEs are essentially modeling infintesimal steps of a residual block (He et al., 2016; Chang et al., 2018), and Grathwohl et al. (2019) found that weight decay & spectral normalization (Miyato et al., 2018) are useful (though expensive) in reducing the rapidly growing number of functional evaluations (NFEs) needed to solve for the ODE endpoint. On the other hand, large-scale DEQ networks (Bai et al., 2019; 2020) have adopted weight normalization (Salimans & Kingma, 2016), recurrent dropout (Gal & Ghahramani, 2016), and group normalization (Wu & He, 2018) for preventing overfitting and divergence. Nonetheless, all these methods are borrowed from explicit deep networks, where they have long been known to work well. They do not exploit the implicitness of implicit models.

More recently, a few different regularization methods have been introduced to specifically fix the numerous issues of the vanilla Neural ODE and continuous normalizing flow models, such as augmenting the hidden state (Dupont et al., 2019), using hyper ODE solvers (Poli et al., 2020), and regularizing higher-order time derivatives (Kelly et al., 2020). These methods directly leverage the dynamical system view of Neural ODEs. However, due to the inherent challenge of solving high-dimensional ODEs, these accelerated Neural ODE models can still easily take > 100 forward iterations even on MNIST classification (Kelly et al., 2020), and even more for their backward pass. In comparison, DEQs scale better to high-dimensional tasks (e.g., 25-30 iterations on



(a) Without regularizations, the relative residual of a DEQ's final output gets worse over training. Both models achieve roughly the same eventual level of accuracy.

(b) In the same setting as Figure 1a, DEQ's convergence residual vs. Jacobian norm $\|J_f\|_F^2$.

(c) Perplexity (ppl) of DEQs on WikiText-103 language modeling as a function of time.

Figure 1. Visualizations of DEQs' instablity and inefficiency problems.

ImageNet) (Bai et al., 2020) and complex f_θ (e.g., a Transformer layer). But such extra complexities also make DEQ models harder to regularize; e.g., simply resorting to weight decay doesn't fix the instability of DEQs (see Section 5.5). To the best of our knowledge, there has been almost no exploration of directly regularizing DEQ stability and convergence.

Our method is closely connected to the many prior works that study Jacobian/gradient regularization (Drucker & Le Cun, 1992; Novak et al., 2018; Hoffman et al., 2019; Finlay et al., 2020; Linsley et al., 2020), though these were also motivated differently. Specifically, Sokolić et al. (2017); Hoffman et al. (2019) regularized the input-output Jacobians of the entire (very deep) explicit classification networks to increase the prediction margin in a robust learning setting (and are thus expensive). Finlay et al. (2020) was inspired by a kinetic energy view and possible overfitting of a training-time dynamical system. The method of Linsley et al. (2020) targeted (for a Jacobian J) a Lipschitzness level λ, used $\max_i(\mathbf{1}^\top J)_i$ to approximate the matrix 1-norm, and proposed loss $L = \|(\mathbf{1}^\top J - \lambda)^+\|_2$. Yet this approximation is in fact problematic, as it does not provably bound the spectral radius (i.e., stability) at all. For example, matrix

$$J = \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix}$$

has $L = 0$ and yet an eigenvalue of 4 (we also empirically verify that this method does not help DEQ models, exactly due to this issue).

In contrast to these works, the key contributions of our paper are that (1) we provide a thorough discussion & summary of various issues with DEQ models, and how ill-conditioned Jacobians are related to the forward/backward instabilities, via the new lens of fixed-point convergence; and (2) we demonstrate how regularizing the Jacobian of DEQs at the equilibrium point (i.e., the final output $\mathbf{z}^\star$) can provably bound the stability of the forward and backward convergences, thereby addressing these various problems. For example, our experiments show that we can significantly stabilize DEQs with new (and more unstable) architectural variants and accelerate DEQs to be nearly as fast as certain explicit architectures (e.g., we only need ≤ 6 NFEs on CIFAR-10) on tasks across different scales and with comparable accuracy.

3. DEQ Models and Their Discontents

Despite the DEQ models' success in some very challenging tasks, such as Cityscapes semantic segmentation (Cordts et al., 2016; Bai et al., 2020), these models suffer from multiple serious downsides. In this section, we provide a summary of some of these problems. While these issues directly lead to our subsequent discussion on the need for regularization (see Section 4), we also believe such systematic discussion provides a useful overview for potential future research on further addressing these issues.

3.1. Growing Instability during Training

Although a DEQ network has no "depth", a relevant measure of computational efficiency is the number of function evaluations (NFEs) of the layer $f_\theta(\mathbf{z}; \mathbf{x})$ used by the iterative root solver (e.g., Broyden's method (Broyden, 1965)).

However, one common phenomenon to all prior works on DEQs is that the fixed points are increasingly harder to solve for over the course of model training. In other words, as a DEQ's performance gradually improves during training, the NFE required to converge to the same threshold ε (e.g., 10^{-3}) rapidly grows. This observation has been made on different instantiations of equilibrium networks, and regardless of whether the model is provably convergent or not (e.g., (Bai et al., 2019; Winston & Kolter, 2020), where a DEQ at the end of training can take $> 3\times$ more iterations). Intuitively, such tendency to approach "critical stability" implicitly characterizes an inclination of the model to learn

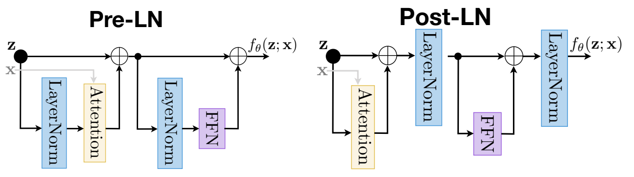


Figure 2. Pre- vs. post-LN DEQ-Transformer layer (Xiong et al., 2020). FFN is a 2-layer feed-forward block (Vaswani et al., 2017).

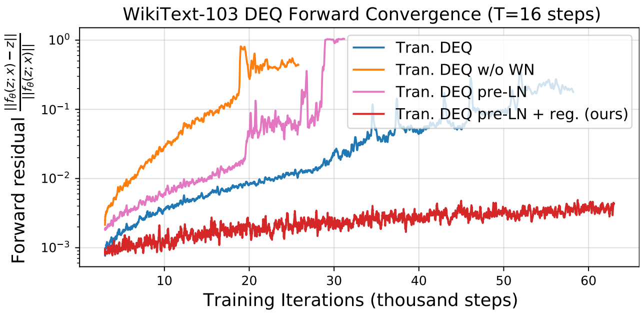
"deeper" networks; so it is unsurprising that unregularized training will keep driving it in this direction. But as a result, the dynamical system only becomes more and more brittle. The existing way of "addressing" this is to circumvent it by setting a maximum NFE limit besides the ε-threshold; i.e., the solver stops either when 1) the residual is smaller than ε, or 2) it has run for a max number of steps T. This could be risky because as the convergence gets more unstable/critical, such a hard stop for the solver cannot guarantee that we are close enough to the fixed point. In the backward pass, for instance, we may consequently be training DEQs with very noisy gradients. A similar issue exists for Neural ODEs, though these cannot easily be hard-stopped like DEQs due to the need to accurately trace the flow to the endpoint.

We illustrate this issue on CIFAR-10 classification in Fig. 1a. One can easily see that both forward and backward estimates of the fixed points gets increasingly worse with the training steps (and eventually plateaus in an unstable region where the model keeps yielding bad gradients). Such growing instability is also reflected empirically in the growth of Jacobian norm at equilibrium; i.e., $\left\|\frac{\partial f_\theta(\mathbf{z}^\star;\mathbf{x})}{\partial \mathbf{z}^\star}\right\|_F$ (see Figure 1b), which we discuss in Section 4. Moreover, interestingly, while these plots might suggest simple regularizations like weight decay, we show later that weight decay often makes this stability issue worse for equilibrium networks, and even leads to divergence.

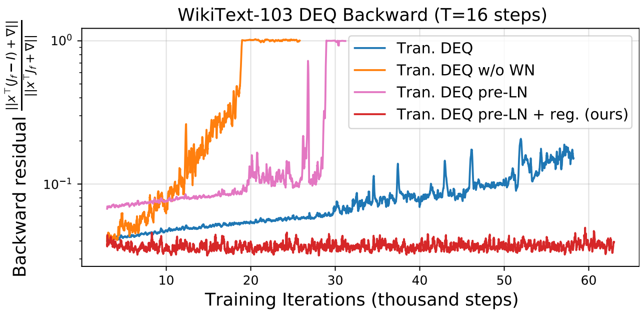
3.2. Inefficiency Compared to Explicit Networks

A direct ramification of the increase in iterations required (see Section 3.1) is the significant increase in both training and inference time for DEQ models.

One advantage of DEQs noted by Bai et al. (2019) is that the forward trajectory need not strictly reach the equilibrium. Therefore in a certain sense, we could trade performance for efficiency by stopping at a "good enough" estimate of the equilibrium. However, due to the growing instability problem, this could still be increasingly costly. This causes the existing DEQs to be significantly slower than their explicit network counterparts of comparable size and performance. E.g., a DEQ-Transformer (Bai et al., 2019) is about 3× slower than a deep Transformer-XL (Dai et al., 2019); a multiscale DEQ (Bai et al., 2020) is over 4× slower than ResNet-101 on ImageNet. Despite their memory efficiency, such slowdown is a roadblock to wider deployment of this class of models in practice. In Figure 1c, we visualize this slowdown on the validation set of WikiText-103 language modeling (Merity et al., 2017) (with comparable model sizes and the same number of training steps).



(a) Forward final objective of fixed-point convergence



(b) Backward final objective of fixed-point convergence

Figure 3. Comparing different architectural modifications of a DEQ-Transformer (first 60K steps). The DEQ networks are brittle: even slight modifications such as changing the whereabouts of LayerNorm (see Figure 2) or removing weight normalization can cause the model to quickly diverge during training.

3.3. Brittleness to Architectural Choices

The need to have a relatively stable DEQ in order to train it via the implicit function theorem also calls for more careful attention in designing the layer f_θ. For example, the largest-scale DEQs (Bai et al., 2019; 2020) all had normalizations (Ba et al., 2016; Wu & He, 2018) at the end of the layer to constrain the output range. How important are these architectural choices? We demonstrate the brittleness of DEQs by ablative studies on the use of layer normalization (LN) or weight normalization (WN) in the DEQ-Transformer model on the large-scale WikiText-103 language modeling task. Specifically, we compare the use of the two most popular Transformer layer designs in the DEQ framework: *pre-LN* and *post-LN*, which simply inserts the LN layers at different parts of the block (see Figure 2). These two settings have been extensively studied, used, and compared in the literature (Liu et al., 2020; Xiong et al., 2020; Vaswani et al., 2017; Baevski & Auli, 2019).

The result is shown in Figure 3. Without layer normalization at the end (magenta line), the DEQ quickly diverges after 25K training iterations (reflected in both forward and backward divergences). Similarly, without weight normalization (orange line), the model becomes unstable more quickly, with fixed-point solver collapse at around 18K iterations. The original DEQ-Transformer (Bai et al., 2019) (blue line in Figure 3), although not diverged, still suffers from the same increased instability problem as described in Section 3.1. These plots are strong indicators that while equilibrium networks work on large scales, they are also relatively inflexible, brittle, and reliant on meticulous architectural designs.

3.4. The Hidden Cost of the Choice of Solver

Although DEQ models enjoy constant memory consumption during training time and can use any black-box fixed point solvers in the forward and backward passes, a commonly neglected cost is that introduced by the choice of solver. For example, in Broyden's method (Broyden, 1965) which Bai et al. (2019; 2020) used, the inverse Jacobian J^{-1} is approximated by low-rank updates of the form $J^{-1} \approx -I + \sum_{i=1}^{n} \mathbf{u}^{[n]}\mathbf{v}^{[n]^\top} = -I + UV^\top$. As another example, Anderson mixing (Anderson, 1965) stores and uses the past m iterations $(\mathbf{z}^{[n-1]}, \dots, \mathbf{z}^{[n-m]})$. In most such cases, even storing these updates or past steps can be expensive. Moreover, since we depend on the same DEQ solvers also at inference time, we need to spend this same memory cost even when when the trained model is served – which conventional deep networks can avoid. We note that this cost depends strongly on the solver; for example, the simplest iterative "solver" $\mathbf{z}^{[i+1]} = f_\theta(\mathbf{z}^{[i]}; \mathbf{x})$ wouldn't have any memory cost, but suffers from bad convergence. This issue also highlights the value of faster and stabler convergence, which entails less memory storage overall (e.g., fewer Broyden steps).

4. Regularizing the Jacobian of DEQs

We hypothesize that one of the fundamental factors contributing to some of the problems discussed in Section 3 is that DEQ models' conditioning is not properly regularized during training. Such trend for DEQ models to go unstable is reflected in Figures 1a and 1b, where increasing training steps leads to monotonically growing residual difference and the Jacobian norm at the equilibrium. We now describe how the Jacobian is related to the stability of equilibrium networks' forward and backward passes, and then harness this relationship to stabilize and accelerate DEQs.

4.1. The DEQ Jacobian

We first recall that the forward pass of a DEQ network aims to solve for the fixed-point representation $\mathbf{z}^\star$ of a layer $f_\theta(\cdot; \mathbf{x})$; i.e., $\mathbf{z}^\star = f_\theta(\mathbf{z}^\star)$. Then in the backward pass, one can differentiate directly through the equilibrium $\mathbf{z}^\star$ by

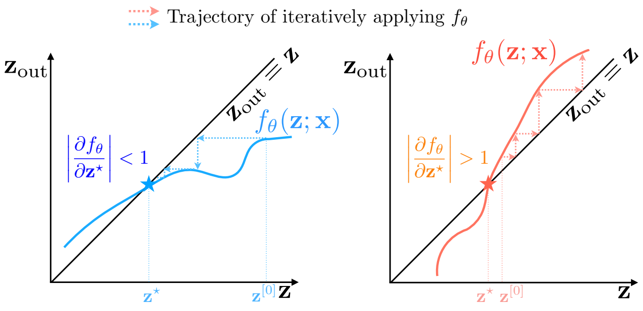

Trajectory of iteratively applying f_θ
$\mathbf{z}_{\text{out}}$
$\mathbf{z}_{\text{out}} = \mathbf{z}$
$f_\theta(\mathbf{z}; \mathbf{x})$
$\left|\frac{\partial f_\theta}{\partial \mathbf{z}^\star}\right| < 1$
$\mathbf{z}^\star$
$\mathbf{z}^{[0]}$
$\mathbf{z}$
$\mathbf{z}_{\text{out}}$
$f_\theta(\mathbf{z}; \mathbf{x})$
$\mathbf{z}_{\text{out}} = \mathbf{z}$
$\left|\frac{\partial f_\theta}{\partial \mathbf{z}^\star}\right| > 1$
$\mathbf{z}^\star$
$\mathbf{z}^{[0]}$
$\mathbf{z}$

Figure 4. Left: when the slope is less than 1, even the simplest iterative application of f_θ converges. Right: when slope > 1, the iterative approach may diverge or oscillate, but the fixed point still exists and can be solved for.

$$\frac{\partial \ell}{\partial (\cdot)} = \underbrace{\frac{\partial \ell}{\partial \mathbf{z}^\star}\left(I - J_{f_\theta}(\mathbf{z}^\star)\right)^{-1}}_{\mathbf{u}^\top} \frac{\partial f_\theta(\mathbf{z}^\star; \mathbf{x})}{\partial (\cdot)}. \tag{1}$$

However, because the scale of J_{f_θ} can be prohibitively large and the inverse is costly to compute, we usually compute the $\mathbf{u}^\top$ term in Eq. 1 by solving the following linear fixed-point system that depends on the final Jacobian:

$$\mathbf{u}^\top = \mathbf{u}^\top J_{f_\theta}(\mathbf{z}^\star) + \frac{\partial \ell}{\partial \mathbf{z}^\star}. \tag{2}$$

Consider the spectral radius of the Jacobian $J_{f_\theta} \in \mathbb{R}^{d \times d}$ at the equilibrium:

$$\rho(J_{f_\theta}(\mathbf{z}^\star)) = \rho(J_{f_\theta}(\mathbf{z}^\star)^\top) = \max(|\lambda_1|, \dots, |\lambda_d|),$$

where λ_is are eigenvalues. In both the forward and backward passes, this spectral radius directly affects how stable the convergence to the fixed point $\mathbf{z}^\star$ could be in its neighborhood. For instance, in the extreme case where we have a contractive $\rho(J_{f_\theta}) < 1$, by Lyapunov linearization theorem even the simplest iterative calls to $f_\theta(\mathbf{z})$ (in forward, assuming good initial estimate) or $g(\mathbf{u}) = \mathbf{u}^\top J_{f_\theta}(\mathbf{z}^\star) + \frac{\partial \ell}{\partial \mathbf{z}^\star}$ (in backward) could converge uniquely, even without advanced solvers. The linear system (2), in particular, would enjoy global asymptotic stability. However in practice, we don't always, and probably shouldn't, require such a strong contractivity on the dynamical system, which might significantly limit the representational capacity of the model. For example, as shown in Figure 4, a fixed point can exist even if $\rho(J_{f_\theta}) > 1$, (the curve slope in 2D); and we are still able to solve for them using the much stronger root solvers (e.g., Newton or quasi-Newton) than these simplest iterative stackings, which could oscillate or diverge.

4.2. Jacobian Regularization

These connections between $J_{f_\theta}(\mathbf{z}^\star)$ (which characterizes the shape of the transformation f_θ around $\mathbf{z}^\star$) and the forward/backward pass dynamics of DEQs motivate us to append a soft and auxiliary Jacobian term $\rho(J_{f_\theta}(\mathbf{z}^\star))$ to the training objective in order to regularize the model's conditioning. One way of doing this is by spectral normalization, essentially constraining $\sigma(J_{f_\theta}) = \max_{\|\mathbf{v}\|\leq 1} \|J_{f_\theta}\mathbf{v}\|_2$. However, explicitly writing out the huge Jacobian and then decomposing it (e.g., by SVD) can be computationally prohibitive, and Miyato et al. (2018) proposes to use the power method (von Mises & Pollaczek-Geiringer, 1929) to speed up this estimation on GANs. But in the context of DEQs, even power iterations are too expensive due to the successive vector-Jacobian product computations needed. Instead, we propose to regularize the Jacobian through its Frobenius norm since

$$\rho(J_{f_\theta}) \leq \sigma(J_{f_\theta}) \leq \sqrt{\text{tr}(J_{f_\theta} J_{f_\theta}^\top)} = \|J_{f_\theta}\|_F.$$

Importantly, $\|J_{f_\theta}\|_F$ can be approximated via various unbiased estimators (Hutchinson, 1989; Ubaru et al., 2017; Meyer et al., 2021). We adopt the classical Hutchinson estimator (Hutchinson, 1989); formally, for $J_{f_\theta} \in \mathbb{R}^{d\times d}$,

$$\text{tr}(J_{f_\theta} J_{f_\theta}^\top) = \mathbb{E}_{\epsilon \in \mathcal{N}(0, I_d)}[\|\epsilon^\top J_{f_\theta}\|_2^2], \tag{3}$$

which we can approximate by Monte-Carlo estimation (i.e., sampling M i.i.d. $\epsilon_i \in \mathcal{N}(0, I_d)$). Specifically, prior works (Avron & Toledo, 2011; Roosta-Khorasani & Ascher, 2015) have established that the relative error of this estimation diminishes with $M^{-\frac{1}{2}}$; and if we compute the mean estimation over a mini-batch size B, the overall relative error with respect to $\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}), \epsilon \in \mathcal{N}(0,I_d)}[\|\epsilon^\top J_{f_\theta}\|_2^2]$ is expected to further diminished by a factor of $B^{-\frac{1}{2}}$ (Hoffman et al., 2019).

Indeed, empirically, we find that $M = 1$ already works well since we use relatively large batch sizes. Since our backward iterations already involved computing multiple vector-Jacobian products $\mathbf{u}^\top J_{f_\theta}$ (see Eq. (2)), computing Eq. (3) only adds a cost equivalent to that of $M = 1$ backward steps. The eventual training objective is thus

$$\mathcal{L}_{\text{total}}(\mathbf{z}^\star) = \mathcal{L}_{\text{orig}}(\mathbf{z}^\star) + \gamma \frac{\|\epsilon^\top J_{f_\theta}(\mathbf{z}^\star)\|_2^2}{d}, \quad \epsilon \in \mathcal{N}(0, I_d) \tag{4}$$

As we observed in Figure 1a, without regularization, a DEQ model that stops after a fixed number T of solver iterations exhibits increasingly poor convergence, accompanied by a growing $\|J_{f_\theta}\|_F$ at these fixed points that empirically signals the growing instability. Therefore, by constraining the Jacobian's Frobenius norm, we encourage DEQs to optimize for stabler and simpler dynamics whose fixed points are easier to solve for.

4.3. Memory Considerations

Although the loss objective (4) only adds minimal computation cost, the need to backpropate through $\|\epsilon^\top J_{f_\theta}\|_2^2$ means we also spend more memory during training to store the computation graph of this vector-Jacobian product. But at the same time, our hidden memory cost due to the solver choice is smaller (e.g., Broyden's method; see Section 3.4) as we can lower the number of iterations. As a result, empirically we notice a roughly 30% net growth in memory consumption compared to the unregularized DEQs at training (and thus saving 50%-60% memory compared to explicit deep networks). The regularized DEQ still consumes $O(1)$ memory relative to the "depth" of the model, as the backpropagation depends only on $\mathbf{z}^\star$.

5. Experiments

We validate the proposed regularization of DEQ models on multiple fronts. First, we visualize the effect of the proposed Jacobian regularization on a tiny DEQ trained on a synthetic 1D dataset. Second, importantly, we focus on how our method alleviates some of the core problems with DEQs outlined in Section 3. Then we show that our method scales to challenging high-dimensional tasks: word-level language modeling with the WikiText-103 dataset (Merity et al., 2017) and image classification with CIFAR-10 and ImageNet (Deng et al., 2009). We specifically compare our model with both prior DEQ networks and competitive explicit models (e.g., ResNet-101, Transformers), in terms of both efficiency (in space and time) and performance. We also explore how Jacobian regularization helps stabilize DEQs over a wider range of architectural choices. Lastly, we perform some ablative studies.

The set of tasks used in our experiment is built directly on top of Bai et al. (2019; 2020). As we found the Jacoabian regularization could sometimes hurt performance (see Sec. 5.3), we only apply the proposed loss stochastically with a probability p, and gradually increase this p or the regularization strength γ (see Eq. (4)) over training steps. We also use cosine learning rate schedule (Loshchilov & Hutter, 2017) for all tasks, including the synthetic one. The memory and speeds reported are benchmarked across different models on the same setting (e.g., same batch size, sequence length, number of steps, etc.) with the same GPU. We provide more details regarding the tasks, hyperparameters, datasets, and hardware in Appendix A, and extra experimental results in Appendix B. Our code and pretrained models are provided here.

5.1. Visualization with Synthetic Data

We start by empirically verifying the validity of the approach and visualizing its effect on a synthetic dataset. We generated 5096 scalar data pairs (x, y) using function $y = h(x) = \frac{3}{2}x^3 + x^2 - 5x + 2\sin(x) - 3 + \delta$ (where $\delta \in \mathcal{N}(0, 0.05)$), and split them into 4096 and 1000 training and validation samples, respectively. We then train a tiny

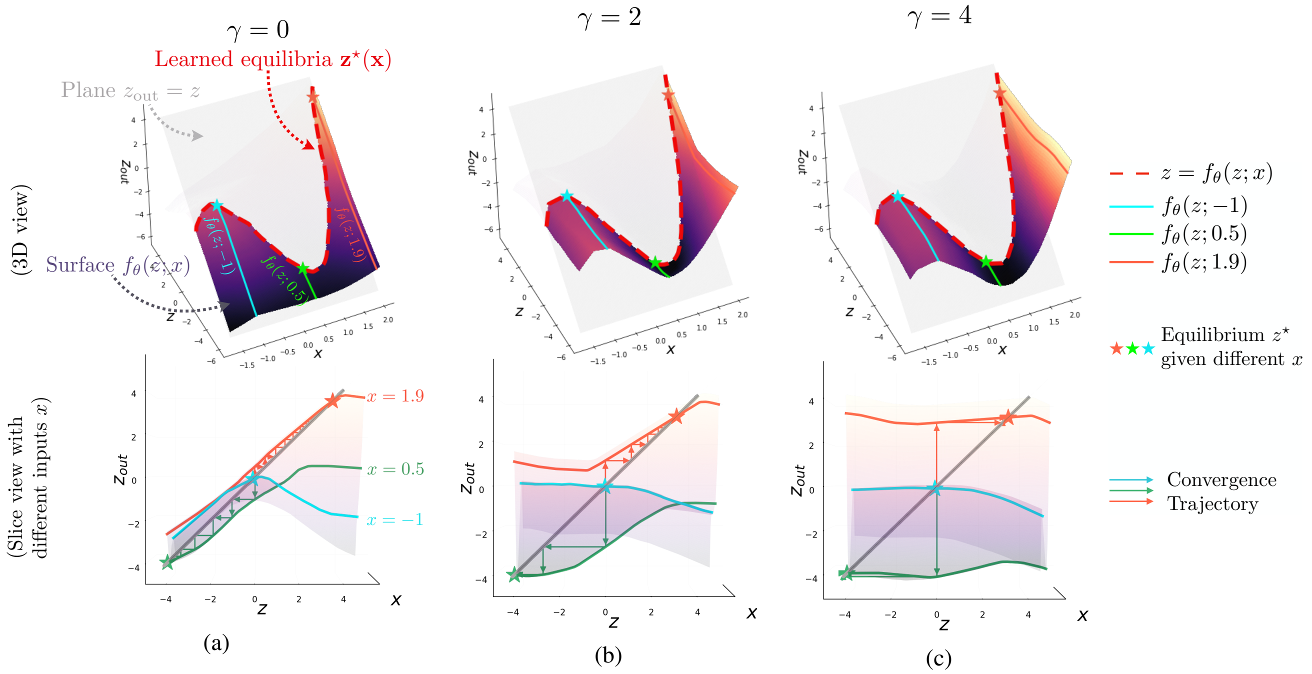


Figure 5. **Top:** the surface of the $f_\theta(\mathbf{z}; \mathbf{x})$ layer, and the eventual learned equilibria $z^\star(x)$ as a function of x. As γ grows, the surface is "lifted up" and becomes flat in the z-direction. **Bottom:** each unique input x defines a slice of the surface, and we perform fixed-point solving on this slice; larger γ values flatten the curve and significantly accelerate the convergence to equilibrium.

DEQ with 200 parameters with the following structure:

$$f_\theta(\mathbf{z}; \mathbf{x}) = W_2^\top \text{ ReLU}(W_1\mathbf{z} + U\mathbf{x} + b), \qquad \hat{y} = \mathbf{z}^\star$$

where we used $\mathbf{z}, \mathbf{x} \in \mathbb{R}$ and $W_1, W_2, U \in \mathbb{R}^{50\times 1}$. The visualizations of the effect of the Jacobian regularization, with different weights γ, are shown in Figure 5. In particular, each input x defines a slice (i.e., cross-section) of the 3D surface $z_{\text{out}} = f_\theta(z; x)$; for example, layer $f_\theta(z; x)$ when input $x = -1$ is highlighted in blue. After training, all three settings succesfully learned the (almost) identical equilibrium function $z^\star(x)$ (highlighted by the red dashed line) that perfectly fits the target function $h(x)$; but note that surfaces of f_θ with $\gamma = 2, 4$ are "lifted up" significantly compared to the unregularized ($\gamma = 0$) DEQ, which has a steep slope (i.e., large spectral radius in 2D). This slope slows down the fixed-point convergence, as reflected by the zigzag patterns in lower Figure 5a. In contrast, the convergences for the $\gamma > 0$ cases are much faster, and larger γ typically yields flatter surfaces around the equilibrium point.

5.2. WikiText-103 Language Modeling

One of the very first successes of large-scale DEQs was its Transformer instantiation (Bai et al., 2019), which uses a multi-head self-attention (Vaswani et al., 2017) layer as the underlying $f_\theta(\mathbf{z}; \mathbf{x})$ function. Although a DEQ-Transformer is able to perform competitively with a deep Transformer-XL (Dai et al., 2019) in terms of test perplexity, and consumes 60-70% less memory, it is also much slower (about $3\times$; see Figure 1c) and borders on instability. In Table 1, we demonstrate how the Jacobian regularization alleviates this.

Table 1. Evaluation on WikiText-103. PPL stands for Perplexity. All Transformer models are trained for 250K steps. t_{train} stands for relative training time. **JR** stands for Jacobian regularization. NFEs are measured at inference time. † indicates unregularized model hard-stopped at inference time.

	Size	PPL	NFEs	t_{train}
AWD-QRNN (Bradbury et al., 2017)	159M	33.0	-	-
Rel. Memory Core (Santoro et al., 2018)	195M	31.6	-	-
18L-Transformer-XL (Dai et al., 2019)	110M	24.1	-	$1\times$
DEQ-Trans. (Pre-LN) (Bai et al., 2019)	98M	[div.]	30	$3.1\times$
DEQ-Trans. (Post-LN) (Bai et al., 2019)	98M	24.0	30	$3.1\times$
DEQ-Trans. (Post-LN) *early stopped*†	98M	29.2	12†	$3.1\times$
DEQ-Trans. (Pre-LN) + JR (ours)	98M	24.5	**14**	$1.6\times$
DEQ-Trans. (Post-LN) + JR (ours)	98M	24.9	**12**	$1.5\times$

Compared to the original DEQ models, there are two major improvements. First, we significantly reduce the NFEs required for DEQ-Transformer models while maintaining competitive accuracy. Using the Transformer-XL as a time benchmark ($1\times$), the speed of a DEQ-Transformer is significantly accelerated: training time goes from $3.1\times$ to $1.5\times$. Second, the regularized DEQ model is more flexible with architectural choices. Whereas a Pre-LN DEQ-Transformer (see Figure 2) quickly diverges in training even in the presence of a large NFE threshold, the Jacobian regularization resolves this issue and stabilizes the forward/backward convergences consistently (see Figure 3 and Table 1), eventually reaching 24.5 perplexity. Moreover, while we can early-stop a well-trained unregularized DEQ model at inference time, it hurts generalization performance significantly (e.g., 29.2

Table 2. Results on CIFAR-10 and ImageNet classfication. The CIFAR-10 accuracy standard deviation is calculated with 5 runs. **JR** stands for Jacobian regularization. † indicates unregularized model hard-stopped at inference time.

CIFAR-10 classification	Size	Accuracy	NFEs
ResNet-18 (He et al., 2016)	10M	93.0 (± 0.1)%	-
ResNet-101 (He et al., 2016)	40M	93.8 (± 0.3)%	-
DenseNet-121 (Huang et al., 2017)	8M	95.0 (±0.1)%	-
monotone DEQ (Winston & Kolter, 2020)	1M	89.4 (± 0.2)%	24
MDEQ (Bai et al., 2020)	10M	93.6 (± 0.2)%	17
MDEQ *early stopped*†	10M	89.1%	6†
MDEQ + JR (ours) (Bai et al., 2020)	10M	93.1 (± 0.3)%	**6**

(Full) ImageNet classification	Size	Top-1 Acc.	NFEs
ResNet-18 (He et al., 2016)	13M	70.2%	-
Inception-V2 (Ioffe & Szegedy, 2015)	12M	74.8%	-
ResNet-50 (He et al., 2016)	26M	75.1%	-
ResNet-101 (He et al., 2016)	52M	77.1%	-
DenseNet-264 (Huang et al., 2017)	74M	79.7%	-
MDEQ-small (Bai et al., 2020)	18M	75.4%	27
MDEQ-large (Bai et al., 2020)	63M	77.5%	30
MDEQ-small + JR (ours)	17M	74.5%	14
MDEQ-large + JR (ours)	62M	76.8%	15

ppl with 12 NFEs). Similarly, we find training with NFEs < 30 leads to increasingly bad generalization performance, and when NFEs drops below 20, model training frequently diverge as a result of extremely noisy gradients. We provide more comprehensive results in Table 5 in the Appendix.

Like DEQs, the regularized DEQs are memory efficient, consuming about 45% less training memory than Transformer-XL. Moreover, we find the Jacobian-regularized DEQs reduce over 50% memory consumption of the original DEQs at inference time (when both using Broyden's method) due to faster/stabler convergence, suggesting its effectiveness in addressing the hidden solver cost issue discussed in Sec. 3.4.

5.3. CIFAR-10 and ImageNet Classification

We additionally conduct experiments on vision tasks using the recent multiscale deep equilibrium networks (MDEQ) (Bai et al., 2020), which drive multiple feature resolutions to their equilibria simultaneously. Because of the need to maintain high- and low-resolutional feature maps at all iterative steps and generally higher channel dimensions in f_θ, MDEQs are substantially slower than conventional networks like ResNets (which operate on progressively downsampled feature maps). This makes acceleration vital to broader adoption of multiscale implicit models.

The results of applying Jacobian regularization on multiscale DEQs for image classification are shown in Table 2. On CIFAR-10, whereas the unregularized DEQ models used 17 NFEs to reach the reported competitive level of performance, our DEQ with Jacobian regularization can converge well even within 6 iterations (in fact, we find smaller NFE values still trains, but significantly hurts generalization performance). This improvement is also obvious in Figure 1a and 1b, where we show that early stopping at threshold $T = 6$ still yields good convergence with Jacobian regularization. We also demonstrate a more stable backward pass convergence throughout training in Appendix B. On the much larger-scale ImageNet, where we deal with 224×224 images, the factor of reduction in NFE is not as strong (e.g., from 27 to 14 iterations, due to the receptive field issue; we'll explain this in Section 5.5) but still yields a roughly $2\times$ acceleration. This shows that the Jacobian regularization is effective in large-scale computer vision tasks, and in the presence of multiple equilibrium points. However, we also note that as with DEQ-Transformers on WikiText-103, we notice a small slip in accuracy, which may be a result of constraining model parameterizations.

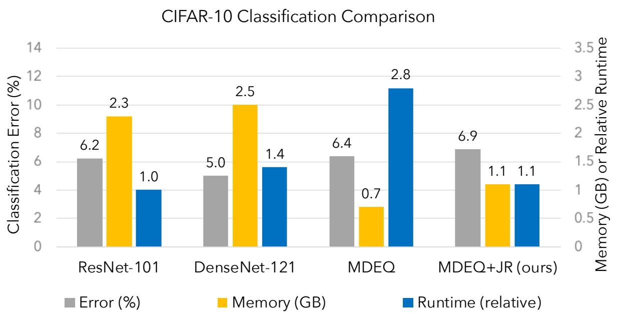


Figure 6. With the proposed regularization, DEQ models are competitive with popular explicit networks in accuracy, memory, and runtime. **Lower bars are better.**

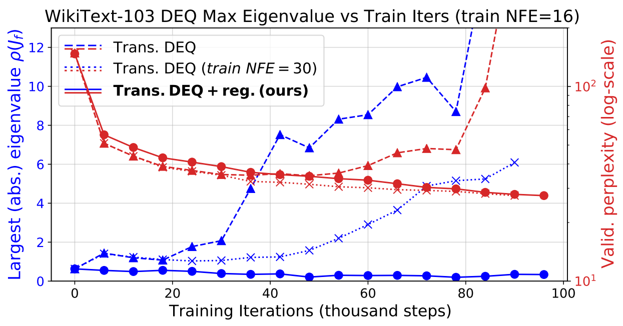


Figure 7. Empirical evidence of how our method constrains $\rho(J_{f_\theta})$ (estimated by power method). In contrast, insufficient NFEs (e.g., T=16) at training time cause a DEQ-Transformer model to explode early in the training phase.

Figure 6 provides a visual comparison of different models with respect to three metrics: performance, inference speed, and training memory. These are reported on the CIFAR-10 dataset. For the first time, we have an implicit-depth model that runs with a competitive level of speed and accuracy as large explicit networks such as ResNet-101, while consuming much less memory.

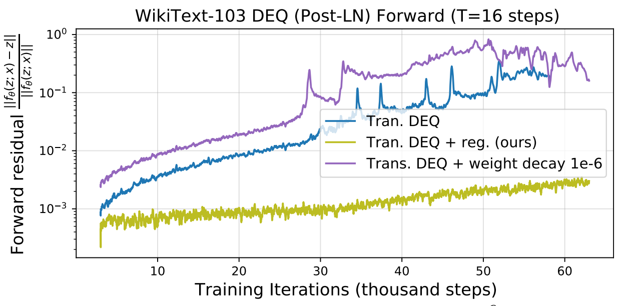


Figure 8. Adding weight decay of magnitude 10^{-6} to the DEQ-Transformer doesn't help stabilize the forward convergence.

Table 3. Controlled experiments on the strength γ of the Jacobian regularization. The NFE value represents the "hard stop" threshold we set for the corresponding DEQ models at inference.

	NFE=1	NFE=2	NFE=3	NFE=4	NFE=5	NFE=6
$\gamma = 0.1$	82.4%	89.7%	91.9%	92.3%	92.7%	92.9%
$\gamma = 0.6$	**85.8%**	**91.5%**	**92.7%**	**93.0%**	**93.0%**	**93.1%**
$\gamma = 1.2$	84.4%	89.6%	92.2%	92.6%	92.7%	92.7%

5.4. Effect of Jacobian Regularization on $\rho(J_{f_\theta})$

In addition to the synthetic study, we also verify that the Jacobian regularization is indeed effectively constraining conditioning of J_{f_θ}. Note that the underlying Jacobian matrices are large (e.g., $[(B{\cdot}110\text{K}) \times (B{\cdot}110\text{K})]$ in WikiText-103, and $[(B{\cdot}198\text{K}) \times (B{\cdot}198\text{K})]$ in ImageNet with MDEQ-small) and checking their full spectrum would be infeasible. Therefore, we conduct a study that monitors the average spectral radius $\rho(J_{f_\theta}(\mathbf{z}^\star))$ (i.e., the largest absolute eigenvalue) on the validation set, over the first 100K steps of DEQ training on WikiText-103 using the power method (von Mises & Pollaczek-Geiringer, 1929); see Fig. 7. Importantly, although $\|J_{f_\theta}\|_F$ only upper-bounds the spectral radius (see Sec. 4.2), we verify that our proposed regularization does effectively constrain $\rho(J_{f_\theta})$ (see ●/● paths in Fig. 7), thereby making DEQs more stable. In contrast, the unregularized DEQ with the same few NFEs explodes in both eigenvalue and shortly after also in perplexity (see ▲/▲ paths), and only works if we increase NFE to 30 (see ×/× paths). In general, we empirically observe that training an unregularized DEQ with insufficient NFEs generally begets extremely noisy gradients, thus leading to faster destabilization and even divergence.

5.5. Ablative Analysis and Limitations of the Approach

We continue our discussion with some empirical ablative studies. First, while Grathwohl et al. (2019) found weight decay useful for regularizing ODE-based models' NFEs, we found weight decay generally not effective in stabilizing DEQs and sometimes even counter-productive. This is illustrated in Figure 8, where after 50K steps the model started to diverge to > 500 perplexity and stopped improving. In addition, we also conduct an ablative experiment on how the Jacobian regularization strength γ affects the performance when we constrain NFEs to ≤ 6 at inference time, with results shown in Table 3 (CIFAR-10 dataset). In general, we find that if γ is too small, the final performance may be good but entails more NFEs. When γ is too large, the accuracy does quickly converge, but the constraint imposed on the model class is too strong and eventually hurts performance (e.g., since the training loss on CIFAR-10 usually overfits to almost 0 towards the end of training, which makes the Jacobian loss dominant instead).

We also highlight two limitations of this approach. First, the addition of Jacobian regularization term does not fundamentally solve the growing instability problem, but only empirically alleviates it. This means that we have to be careful about balancing the main loss objective and this auxiliary objective (see Table 3). Second, while Jacobian regularization facilitates faster convergence, there are certain "physical laws" that we simply cannot bypass. For example, if we apply a shallow convolutional DEQ whose layer has receptive field 5×5 on a large image (e.g., 1024×1024), it is hard to be able to reach the fixed point with just 6 iterations simply because the model's receptive field may not broaden sufficiently to cover valuable context. Although one can possibly still force convergence with a large γ, it would undoubtedly hurt the performance. This explains why we need more NFEs on ImageNet than on CIFAR-10 (see Table 2); it also indicates that while our approach alleviates the brittleness to architectural choices, its effectiveness can still depend on the architecture. This makes global-context alternatives to ConvNets, such as self-attention-based layers (e.g.,ViT (Dosovitskiy et al., 2020)) likely more appealing in the implicit model setting, which we leave for future work.

6. Conclusion

We summarized the weaknesses of existing DEQ models, including instability & inefficiency, architectural brittleness, and hidden memory costs. We specifically discussed the relationship between the spectral radius of the Jacobian and the stability of forward non-linear and backward linear systems of DEQ models, and provided empirical evidence of the poor conditioning of the Jacobian. This motivates our introduction of Jacobian regularization. Our experiments show that our method significantly alleviates the weaknesses of DEQs, yielding a $> 2.5\times$ acceleration. This is a major step towards making implicit models more practical and suitable for large-scale real-world applications. We hope that our work will motivate further research that advances our understanding and application of this class of models.

References

Amos, B. and Kolter, J. Z. OptNet: Differentiable optimization as a layer in neural networks. In *International*

Conference on Machine Learning (ICML), 2017.

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

Avron, H. and Toledo, S. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *Journal of the ACM (JACM)*, 58(2):1–34, 2011.

Ba, L. J., Kiros, R., and Hinton, G. E. Layer normalization. *arXiv:1607.06450*, 2016.

Baevski, A. and Auli, M. Adaptive input representations for neural language modeling. In *International Conference on Learning Representations (ICLR)*, 2019.

Bai, S., Kolter, J. Z., and Koltun, V. Deep equilibrium models. In *Neural Information Processing Systems*, 2019.

Bai, S., Koltun, V., and Kolter, J. Z. Multiscale deep equilibrium models. In *Neural Information Processing Systems*, 2020.

Bradbury, J., Merity, S., Xiong, C., and Socher, R. Quasi-recurrent neural networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *arXiv:2005.14165*, 2020.

Broyden, C. G. A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation*, 1965.

Chang, B., Meng, L., Haber, E., Tung, F., and Begert, D. Multi-level residual networks from dynamical systems view. In *International Conference on Learning Representations (ICLR)*, 2018.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *Neural Information Processing Systems*, 2018.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. The Cityscapes dataset for semantic urban scene understanding. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. Transformer-XL: Attentive language models beyond a fixed-length context. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Li, F. ImageNet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv:2010.11929*, 2020.

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

Dupont, E., Doucet, A., and Teh, Y. W. Augmented neural ODEs. In *Neural Information Processing Systems*, 2019.

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

El Ghaoui, L., Gu, F., Travacca, B., and Askari, A. Implicit deep learning. *arXiv:1908.06315*, 2019.

Finlay, C., Jacobsen, J.-H., Nurbekyan, L., and Oberman, A. M. How to train your neural ODE. *arXiv:2002.02798*, 2020.

Gal, Y. and Ghahramani, Z. A theoretically grounded application of dropout in recurrent neural networks. In *Neural Information Processing Systems*, 2016.

Gould, S., Hartley, R., and Campbell, D. Deep declarative networks: A new hope. *arXiv:1909.04866*, 2019.

Grathwohl, W., Chen, R. T., Betterncourt, J., Sutskever, I., and Duvenaud, D. FFJORD: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations (ICLR)*, 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

Hoffman, J., Roberts, D. A., and Yaida, S. Robust learning with Jacobian regularization. *arXiv:1908.02729*, 2019.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

Hutchinson, M. F. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015.

Kawaguchi, K. Dynamics of deep equilibrium linear models. In *International Conference on Learning Representations (ICLR)*, 2021.

Kelly, J., Bettencourt, J., Johnson, M. J., and Duvenaud, D. Learning differential equations that are easy to solve. In *Neural Information Processing Systems*, 2020.

Krantz, S. G. and Parks, H. R. *The implicit function theorem: History, theory, and applications*. Springer, 2012.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. In *Neural Information Processing Systems*, 2012.

Linsley, D., Ashok, A. K., Govindarajan, L. N., Liu, R., and Serre, T. Stable and expressive recurrent vision models. *arXiv:2005.11362*, 2020.

Liu, L., Liu, X., Gao, J., Chen, W., and Han, J. Understanding the difficulty of training transformers. *arXiv:2004.08249*, 2020.

Loshchilov, I. and Hutter, F. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017.

Lu, C., Chen, J., Li, C., Wang, Q., and Zhu, J. Implicit normalizing flows. In *International Conference on Learning Representations (ICLR)*, 2021.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2), 1993.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. In *International Conference on Learning Representations (ICLR)*, 2017.

Merity, S., Keskar, N. S., and Socher, R. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations (ICLR)*, 2018.

Meyer, R. A., Musco, C., Musco, C., and Woodruff, D. P. Hutch++: Optimal stochastic trace estimation. In *Symposium on Simplicity in Algorithms (SOSA)*, 2021.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. Sensitivity and generalization in neural networks: An empirical study. *arXiv:1802.08760*, 2018.

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

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

Poli, M., Massaroli, S., Yamashita, A., Asama, H., and Park, J. Hypersolvers: Toward fast continuous-depth models. *arXiv:2007.09601*, 2020.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Revay, M., Wang, R., and Manchester, I. R. Lipschitz bounded equilibrium networks. *arXiv:2010.01732*, 2020.

Roosta-Khorasani, F. and Ascher, U. Improved bounds on sample size for implicit matrix trace estimators. *Foundations of Computational Mathematics*, 15(5):1187–1212, 2015.

Salimans, T. and Kingma, D. P. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Neural Information Processing Systems*, 2016.

Santoro, A., Faulkner, R., Raposo, D., Rae, J., Chrzanowski, M., Weber, T., Wierstra, D., Vinyals, O., Pascanu, R., and Lillicrap, T. Relational recurrent neural networks. In *Neural Information Processing Systems*, 2018.

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catanzaro, B. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv:1909.08053*, 2019.

Sokolić, J., Giryes, R., Sapiro, G., and Rodrigues, M. R. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280, 2017.

Ubaru, S., Chen, J., and Saad, Y. Fast estimation of tr(f(a)) via stochastic lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1075–1099, 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Neural Information Processing Systems*, 2017.

von Mises, R. and Pollaczek-Geiringer, H. Praktische verfahren der gleichungsauflösung. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 9(1):58–77, 1929.

Winston, E. and Kolter, J. Z. Monotone operator equilibrium networks. In *Neural Information Processing Systems*, 2020.

Wu, Y. and He, K. Group normalization. In *European Conference on Computer Vision (ECCV)*, 2018.

Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., and Liu, T. On layer normalization in the transformer architecture. In *International Conference on Machine Learning (ICML)*, 2020.

Accelerating Equilibrium Models by Stabilizing Their Jacobians
Supplementary Material

A. Dataset Information, Experimental Settings and Hyperparameters

We provide below a detailed description of all tasks and settings for experiments reported in Section 5, as well as some training specifics of the deep equilibrium network (DEQs) we use.

A.1. 1D Synthetic Dataset

To visualize the effect of the proposed Jacobian regularization on DEQ models (see Section 5), we generated a synthetic dataset with 5096 pairs (x, y) from the target function:

$$y = h(x) = \frac{3}{2}x^3 + x^2 + 5x + 2\sin(x) - 3 + \delta$$

where $\delta \in \mathcal{N}(0, 0.05)$ are i.i.d. noise variables added to each sample in the dataset. Specifically, we split the generated data into 4096 training samples and 1000 validation samples.

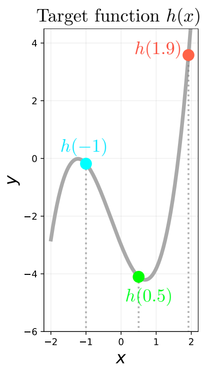


Figure 9. Target function $y = h(x)$.

Figure 9 shows the target function. In the context of deep equilibrium networks, we aim to learn a function $z^\star(x)$ such that $z^\star = f_\theta(z^\star; x)$ and $z^\star(x) \approx h(x)$. At a high level, we should expect the *intersection* between the $z_{\text{out}} = f_\theta(z; x)$ surface and the $z_{\text{out}} = z$ plane to be exactly like the gray curve in Figure 9.

The learned DEQ equilibria $z^\star(x)$ are empirically demonstrated in Figure 5 in red dashed lines for different choices of γ. As expected, all γ fit the target function perfectly, but the introduction of the Jacobian regularization makes the surface more flat around the fixed point.

A.2. WikiText-103 Word-level Language Modeling

Word-level language modeling tasks aim to predict the next word of a textual sequence by integrating the semantics and information of current and past tokens. Formally, given an input sequence $\mathbf{x}_{1:T} \in \mathbb{R}^{T\times p}$ (where $x_i \in \mathbb{R}^p$ and T is the sequence length), an autoregressive sequence model G produces output $G(\mathbf{x}_{1:T}) = \mathbf{y}_{1:T} \in \mathbb{R}^{T\times q}$ that satisfies the causality constraint: y_t depends only on $x_{1:t}$ and not on the future information $x_{t+1:T}$. When each x_i represents a word (i.e., a word embedding), the task is essentially a *word-level language modeling task*. This is a widely-studied problem in the NLP community (e.g., (Merity et al., 2017; 2018; Dai et al., 2019)), and has seen practical advancement in the last few years with development of GPT-3 (Brown et al., 2020; Radford et al., 2019).

A commonly used large-scale corpus for this task is the WikiText-103 (Merity et al., 2017) dataset, which contains 103M/217K/246K words at train/validation/test time, respectively. The entire corpus has a vocabulary size of 267K (i.e., the number of rows in the word embedding). Unlike other well-processed, much smaller datasets like Penn Treebank (Marcus et al., 1993), WikiText-103 is much more challenging as it contains many rare words and retains punctuations, numbers, upper- and lower-cases from the source Wikipedia articles; it has been the standard benchmark for many high-capacity language models in recent literature (Merity et al., 2018; Bradbury et al., 2017; Dai et al., 2019). We provide a shell script in our submitted code to download this dataset.[1]

A.3. CIFAR-10 & ImageNet Image Classification

The CIFAR-10 (Krizhevsky & Hinton, 2009) dataset contains 60,000 color images of resolution 32×32 that fall into 10 object classes (with uniformly 6,000 images per class). We use the standard setting where 50K of these images are used for training and the rest 10K for validation purpose.

The ImageNet (Krizhevsky et al., 2012) dataset, on the other hand, contains over 1.28M training images and 150K test images, distributed over 1,000 classes. All images are rescaled to 224×244 resolution before they are fed into the models (as the original images are of variable resolutions and scales). This is a frequently used dataset for evaluating large-scale vision networks, and has been used for also pretraining many image feature extractor for use on downstream tasks.

For both CIFAR-10 and ImageNet, each training image goes through a canonical data augmentation process before they are fed into the model, where we perform random cropping and random horizontal flipping.

[1]Officially, this dataset can be downloaded at this link.

Table 4. Hyperparameters, optimizer choices, and model details (at training time) for all tasks reported in Section 5. The arrows in the Jacobian regularization strength (e.g., $A \to B$) mean that we dynamically increase from A to B over the course of DEQ training.

	Synthetic Dataset	**WikiText-103 language modeling**	**CIFAR-10 classification**	**ImageNet classification**
Architecture of f_θ	2-Layer ReLU block (see Section 5)	Transformer layer (Pre- and Post-LN)	Multiscale DEQ layer (residual block + fusion)	Multiscale DEQ layer (residual block + fusion)
# of Epochs	50	23	200	120
Batch Size	64	60	96	112
Optimizer	Adam	Adam	Adam	SGD
Start Learning rate	0.001	0.00025	0.001	0.05
Learning rate warmup	No	Yes, 1 epoch	No	No
Learning rate schedule	Cosine	Cosine	Cosine	Cosine
Weight Decay	0	0	0	$5 \cdot 10^{-5}$
Hidden dimensionality	50	700 (embedding size)	[28,56,112,224] (4 scales)	[32,64,128,256]
Input Sequence Length	N/A	150	N/A	N/A
Input Image Size	N/A	N/A	32×32	224×224
Normalization	None	LayerNorm (Ba et al., 2016)	GroupNorm (Wu & He, 2018)	GroupNorm
Recurrent Droput	N/A	0.06	0.25	0.02
Weight Normalization	No	Yes	Yes	Yes
# of Input Injection Downsamplings	N/A	N/A	N/A	2
Fixed-point Solver	Anderson	(Good) Broyden	Anderson	(Good) Broyden
Forward NFEs Threshold	6	12	7	14
Backward NFEs Threshold	6	12	8	14
Forward Threshold ε	10^{-3}	10^{-3}	10^{-3}	10^{-3}
Backward Threshold ε	10^{-4}	10^{-4}	10^{-4}	10^{-4}
Jacobian Reg. Strength γ	{0,1,2,4}	$1.6 \to 2.5$	0.5	$2.0 \to 3.0$
Jacobian Reg. Frequency p	0.4	0.35	0.05	0.1
M for Hutchinson Estimator	1	1 or 2	1	1 or 2

A.4. Training Setting and Hardware

Our experimental protocols are intentionally set to be maximally consistent with prior work (Bai et al., 2019; 2020). This includes hyperparameters (see the subsection below), other regularization methods (e.g., recurrent dropout (Gal & Ghahramani, 2016) & group normalization (Wu & He, 2018)), and initialization schemes (where all parameters are initialized at the start of training by sampling from $\mathcal{N}(0, 0.01)$). For the multiscale DEQs that were used in the image classification task, we used 4 resolutions, where each subsequent resolution is of exactly half the height and width of the previous resolution. Although Bai et al. (2020) highlighted the need to train a ReLU-based network with softplus for stability purposes, we found it not necessary in our experiments with regularized DEQs, most likely because of the role Jacobian regularization plays in stabilizing the network convergence.

One thing to note is that empirically, rather than applying the proposed Jacobian regularization on all training iterations, we only randomly and partially apply this auxiliary loss. For example, when we set the auxiliary loss frequency p to 0.5, only half of the training iterations (randomly selected) are trained with the Jacobian regularization term (see Table 4). This is motivated by the empirical observation that Jacobian-related regularizations usually hurt performance, e.g., as in its application in robust learning (Hoffman et al., 2019). Therefore, such partial/random supervision with the Jacobian regularization brings two benefits: 1) the rest $(1-p)$-portion of the training iterations can pick up a further speedup as we don't need to compute the Hutchinson estimator and backpropagate through it; and 2) it helps reduce the likelihood of the model overfitting on this auxiliary loss term (since, as we noted in Section 5.5, the model could be sensitive to γ, and M is small), which we generally observe to benefit the performance, though only slightly. Therefore, during training, the model would still proceed in the actual stochastic gradient direction, and only use the regularized direction occasionally.

Formally, the training objective we highlighted in Section 4.2 should be:

$$\mathcal{L}_{\text{total}}(\mathbf{z}^\star) = \mathcal{L}_{\text{orig}}(\mathbf{z}^\star) + \tau \cdot \gamma \frac{\sum_{m=1}^{M} \|\epsilon^\top J_{f_\theta}(\mathbf{z}^\star)\|_2^2}{Md}, \;\; \epsilon_m \in \mathcal{N}(0, I_d)$$

where $\tau = \text{Bernoulli}(p)$ is a random variable and M is the number of samples used for Hutchinson estimator.

All experiments in this paper, including the speed and memory benchmarks we provide, were conducted on RTX 2080 Ti GPUs. WikiText-103 language modeling and ImageNet classification models (MDEQ-small) were trained with 4 GPUs in a data-parallel setting.

A.5. Hyperparameters

We report the hyperparameters used at training time in Table 4. Except for those used in the synthetic data and for Jacobian regularization, most of the other hyperparameters were essentially taken from the original DEQ-Transformer (Bai et al., 2019) and MDEQ (Bai et al., 2020) without major modifications. For both Anderson

Table 5. A more complete version of Table 1 with more memory and efficiency comparison. Memory benchmarked on batch size 15 and excludes the embedding layer. † indicates unregularized model hard-stopped at inference time (while still trained with more NFEs). Overall, we find that Jacobian regularization allows us to train and predict with much fewer NFEs, at a relatively small cost in performance.

	Model Size	Perplexity	t_{train} (relative)	Train NFE	Valid. NFE	Training Memory
AWD-Quasi RNN (Bradbury et al., 2017)	159M	33.0	-	-	-	7.1GB
Relational Memory Core (Santoro et al., 2018)	195M	31.6	-	-	-	-
Megatron-LM (Shoeybi et al., 2019) [SOTA]	8300M	10.8	-	-	-	-
Transformer-XL (18-layer) (Dai et al., 2019)	110M	24.1	1×	-	-	9.0GB
DEQ-Transformer (Pre-LN) (Bai et al., 2019)	98M	[diverged]	N/A	30	N/A	N/A
DEQ-Transformer (Post-LN) (Bai et al., 2019)	98M	24.0	3.1×	30	30	3.9GB
DEQ-Transformer (Post-LN) *early stopped*	98M	29.2	3.1×	30	12	3.9GB
DEQ-Transformer (Post-LN) (Bai et al., 2019)	98M	26.0	2.2×	20	20	3.6GB
DEQ-Transformer (Post-LN) (Bai et al., 2019)	98M	[diverged]	N/A	15	N/A	3.6GB
DEQ-Transformer (Pre-LN) + JR (ours)	98M	24.5	1.5×	14	14	4.8GB
DEQ-Transformer (Post-LN) + JR (ours)	98M	24.9	1.4×	13	12	4.8GB
DEQ-Transformer (Post-LN) + JR (ours) (trained on seqlen=300)	98M	23.8	2.2×	13	13	6.5GB

and Broyden fixed-point solvers, we use the relative residual $\frac{\|f_\theta(\mathbf{z};\mathbf{x})-\mathbf{z}\|}{\|f_\theta(\mathbf{z};\mathbf{x})\|}$ as a measure of convergence quality in forward and backward passes. At inference time, we generally reduce the number of NFEs (e.g., cf. Table 4 and Table 1), while the other hyperparameters (e.g., GroupNorm group sizes) are kept the same.

B. Additional Experimental Results

B.1. Memory Consumption

As we noted in Sections 4 and 5, using Jacobian regularization and thus the vector-Jacobian-product-based Hutchinson estimator introduces some extra memory cost at training time due to the need to differentiate w.r.t. the $\|J_{f_\theta}\|_F$ term. Overall, with the same batch size and sequence length, we observe a roughly 25% increase in training memory required (from about 3.9GB to 4.8GB, excluding embeddings). This is less than the memory consumption of a layer, because the reduction in NFEs needed on the other side saves the memory used by the solver (see Section 3.4). However, this memory footprint is still much better than the conventional explicit Transformer-XL model, which consumes about 2× as much GPU memory. With the Jacobian regularization, as we can see, the DEQ models are much more efficient in time complexity than before, while still staying competitive on the space complexity and the performance fronts.

B.2. DEQ's Backward Convergence with Jacobian Regularization (CIFAR-10)

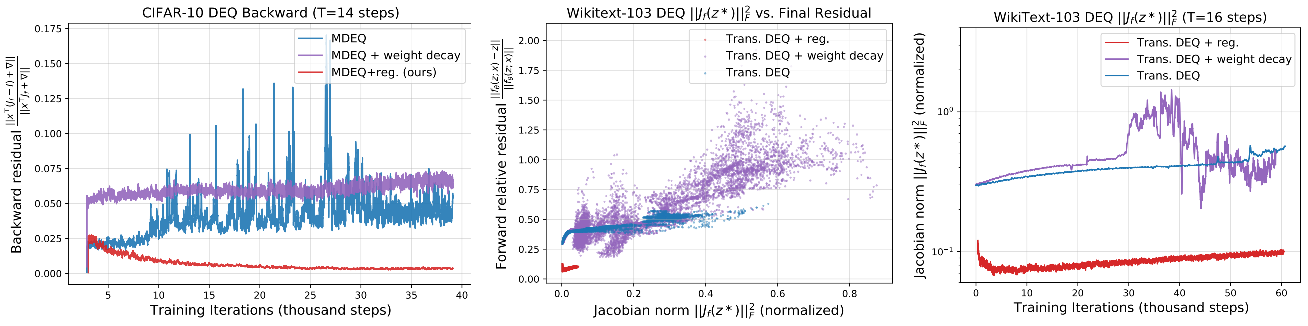
As we discussed in Section 4, the backward dynamics of a DEQ model is a *linear* fixed point system that depends directly on the Jacobian at equilibrium (i.e., $J_{f_\theta}(\mathbf{z}^\star)$). Therefore, the backward pass stability is directly influenced by the conditioning of the Jacobian that we regularize. The stabilizing effect of the proposed Jacobian regularization on the backward pass convergence was already shown for WikiText-103 language modeling in Figure 3b, where we empirically observe that the Jacobian-regularized DEQ-Transformer's backward pass stays at a consistent level, which indicates a relatively more accurate gradient produced by the implicit function theorem.

We further corroborate this finding via empirical evidence on the CIFAR-10 dataset with a multiscale DEQ (MDEQ) instance, shown in Figure 10a. Compared to the original MDEQ (blue line), the Jacobian-regularized version of the backward pass experiences much fewer fluctuations (and thus less stochastic gradients). We also compared to an alternative solution that uses the simple weight decay. Although it also alleviates the fluctuation problem, our empirical observations suggest that weight decay alone almost always adds more difficulty to the fixed point solving. This agrees with what we have observed in the forward pass in Section 5.5. Such comparison can be seen in Figure 10a in the purple line, which converged even more poorly than the original baseline after 14 backward solver iterations (with relative residual > 0.05 and increasing slowly over training). In contrast, the regularized backward pass is more smooth and stable (red line) throughout training (we used $\gamma = 0.5$).

B.3. Failure of Weight Decay to Fix the Problem

This overall inability of weight decay alone to fix the DEQ stability issue (e.g., see Figures 8 and 10a), we believe, exactly suggests that there is a deeper *implicitness* property of the model that should be regularized than just the value of individual weights. As DEQ networks typically rely on a single f_θ block, their complex non-linear structure makes their stability depend as much on the linear parts of f_θ (which weight decay does regularize) as the non-linear parts (which weight decay does not directly regularize; e.g., self-attention in f_θ if we use a Transformer layer). On the other hand, Jacobian regularization takes into account both parts as it tries to constrain the overall spectral radius of the matrix.

We also provide some additional analysis on how $\|J_{f_\theta}\|_F$



(a) Jacobian regularization improves both the fluctuation and quality of the backward convergence.

(b) Forward relative residual on WikiText-103 as a function of the Jacobian norm.

(c) Jacobian norm grows throughout training, even when we regularize for it.

Figure 10. Additional analysis on DEQ models' backward convergence (on CIFAR-10), Jacobian norm, etc.

evolves during training in Figure 10b and 10c. Specifically, even with weight decay, the convergence of DEQ-Transformer models can be quite bad (see purple dots in Figure 10b), with a clear correlation between the larger relative residual and larger $\|J_{f_\theta}\|_F^2$. Indeed, with a non-linear structure as complex as the multi-head self-attention, simply constraining the weights to be small is not sufficient to ensure well-conditioned Jacobians. Moreover, while the Jacobian regularization helps significantly stabilize the forward and backward convergence (see Figure 1a, 10a and 3), we note that a regularized DEQ model still in fact gradually tends to "critical stability". This can be seen in Figure 10c, where the Jacobian norm grows slowly over training iterations (red line) for a fixed γ, though at a rate much slower than the unregularized and weight-decayed baselines. Therefore, as we indicated in Section 5.5, the proposed Jacobian regularization does not fundamentally *fix* the growing instability problem, but only *alleviates* it. This also calls for adaptive γ scheduling during training (which we adopt in a simple form in our implementation and leave more advanced schemes for future work).