A Closer Look at the Adversarial Robustness of Deep Equilibrium Models

Zonghan Yang[1], **Tianyu Pang**[2], **Yang Liu**[1,3,4*]
[1]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[2]Sea AI Lab, Singapore
[3]Institute for AI Industry Research (AIR), Tsinghua University, Beijing, China
[4]Beijing Academy of Artificial Intelligence, Beijing, China
`yangzh20@mails.tsinghua.edu.cn`, `tianyupang@sea.com`, `liuyang2011@tsinghua.edu.cn`

Abstract

Deep equilibrium models (DEQs) refrain from the traditional layer-stacking paradigm and turn to find the fixed point of a single layer. DEQs have achieved promising performance on different applications with featured memory efficiency. At the same time, the adversarial vulnerability of DEQs raises concerns. Several works propose to certify robustness for monotone DEQs. However, limited efforts are devoted to studying empirical robustness for general DEQs. To this end, we observe that an adversarially trained DEQ requires more forward steps to arrive at the equilibrium state, or even violates its fixed-point structure. Besides, the forward and backward tracks of DEQs are misaligned due to the black-box solvers. These facts cause gradient obfuscation when applying the ready-made attacks to evaluate or adversarially train DEQs. Given this, we develop approaches to estimate the intermediate gradients of DEQs and integrate them into the attacking pipelines. Our approaches facilitate fully white-box evaluations and lead to effective adversarial defense for DEQs. Extensive experiments on CIFAR-10 validate the adversarial robustness of DEQs competitive with deep networks of similar sizes.

1 Introduction

Conventional deep networks employ multiple stacked layers to process data in a feedforward manner [17]. During training, network parameters are optimized by backpropagating loss updates through the consecutive layers [36]. Recently, [3] propose deep equilibrium models (DEQs), whose forward pass involves finding the fixed point (i.e., equilibrium state) of a single layer. With implicit differentiation, the backward pass of DEQs is formulated as another linear fixed-point system. Training DEQs with black-box root solvers only consumes $\mathcal{O}(1)$ memory, which enables DEQs to achieve performance competitive with conventional networks in large-scale applications, including language modelling [3], image classification and segmentation [4], density modelling [24, 16], and graph modelling [23].

Considering the fixed point as a local attractor, DEQs are expected to be stable to small input perturbations. However, empirical observations show the opposite that a vanilla DEQ is also vulnerable to adversarial attacks [16]. Along this routine, several works are proposed to investigate the certified robustness for monotone DEQs [40, 34, 27, 20, 28, 10]. Inspired from the monotone operator splitting theories, monotone DEQs are designed with the guarantee of existence and convergence of equilibrium points. However, the layer parameterization of monotone DEQs and the limited scalability of certification methods narrow the scope of these previous studies. On the other hand, [16] explore the adversarial robustness for general DEQs. They incorporate the adversarial generation process into the equilibrium solver to accelerate the PGD attack [25]. Nevertheless, the PGD attack is originally designed for deep networks, requiring for end-to-end white-box differentiation. In contrast, DEQs

[*]Corresponding author: Yang Liu

rely on black-box solvers and could obfuscate the gradients used in PGD: as shown in Fig. 2-(a), in DEQs trained with different configurations, the intermediate states *always* exhibit higher robustness than the final state under *ready-made* PGD attacks. Compared to the extensive literature on the adversarial robustness of deep networks [6, 38, 15, 26, 22, 25, 43, 35, 29], much less is known about the adversarial robustness of general DEQs, especially under a well-elaborate white-box setting. This motivates us to disentangle the modules in DEQs and provide a fair evaluation of their robustness.

In this paper, we first summarize the challenges of training robust DEQs (see Sec. 3), including (**i**) convergence of the black-box solvers and (**ii**) misalignment between the forward and backward passes. The off-the-shelf attacks work in a gray-box setting as they have no access to the intermediate states in the forward pass. To thoroughly evaluate the robustness, we propose two methods for intermediate gradient estimation: the first one is iterating adjoint gradient estimations simultaneously in the forward pass, as formally described in Sec. 4.1; the second one is estimating intermediate gradients by unrolling, as seen in Sec. 4.2. Then in Sec. 5, we develop approaches to integrate the estimated gradients into the ready-made attacks towards fully white-box adversaries. We also design defense strategies for DEQs to boost their robustness under white-box attacks.

We use PGD-AT to train large-sized and XL-sized DEQs on CIFAR-10. To benchmark their robustness [12], the parameter sizes of the DEQs are set to be comparable with ResNet-18 [18] and WideResNet-34-10 [42], respectively. We observe that the adversarially trained DEQs with the exact gradient [3] require more forward steps to arrive at the equilibrium state, or even violate their fixed-point structures. We also find an intriguing robustness accumulation effect that the intermediate states in the forward pass are more robust under ready-made attacks. These phenomena exhibit gradient obfuscation [2], which verifies the necessity of intermediate gradient estimation to construct white-box attacks and defense strategies. Robustness performance under the white-box evaluation shows that DEQs achieve competitive or stronger adversarial robustness than deep networks of similar parameter amounts. Our investigation sheds light on the pros and cons with respect to the adversarial robustness of DEQs.

2 Background

This section includes the background on DEQs and adversarial robustness for deep networks.

2.1 Deep equilibrium models

We first briefly introduce the modelling of deep equilibrium models (DEQs) [3, 4]. Consider a T-layer weight-tied input-injected neural network:

$$\mathbf{z}_n = f_\theta(\mathbf{z}_{n-1}; \mathbf{x}),\ n = 1, \dots, T, \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^l$ is the input, $\mathbf{z}_n \in \mathbb{R}^d$ is the output of the n-th layer, and θ is the network weights shared across different layers. One can cast the evolution of $\{\mathbf{z}_n\}$ as a fixed-point iteration process. When $n \to \infty$, $\mathbf{z}_n$ converges to the fixed point $\mathbf{z}^*$ which satisfies the equation $\mathbf{z}^* = f_\theta(\mathbf{z}^*; \mathbf{x})$.

Deep equilibrium models rely on the fixed-point equation and leverage a black-box solver to *directly* solve for $\mathbf{z}^*$ in the forward pass. The backward pass of DEQs can also be formulated as a fixed-point iteration process. With the loss function $L(\mathbf{z}^*, y)$ and implicit differentiation, we can compute the gradient with respect to θ or $\mathbf{x}$ with

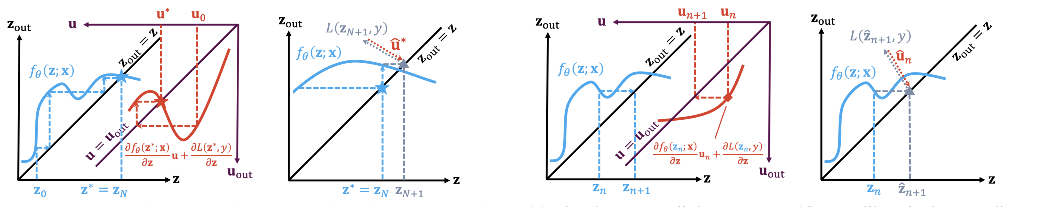
$$\frac{\partial L}{\partial(\cdot)} = \left(\frac{\partial f_\theta(\mathbf{z}^*; \mathbf{x})}{\partial(\cdot)}\right) \underbrace{\left(I - \frac{\partial f_\theta(\mathbf{z}^*; \mathbf{x})}{\partial \mathbf{z}}\right)^{-1} \frac{\partial L(\mathbf{z}^*, y)}{\partial \mathbf{z}}}_{\mathbf{u}^*}, \tag{2}$$

where $(\partial \mathbf{a}/\partial \mathbf{b})_{ij} = \partial \mathbf{a}_j / \partial \mathbf{b}_i$ and $\mathbf{u}^*$ satisfies

$$\mathbf{u}^* = \left(\frac{\partial f_\theta(\mathbf{z}^*; \mathbf{x})}{\partial \mathbf{z}}\right) \mathbf{u}^* + \frac{\partial L(\mathbf{z}^*, y)}{\partial \mathbf{z}}. \tag{3}$$

According to Eq. (3), the backward pass can also be executed with a black-box fixed-point solver, and this iteration process is independent of that in the forward pass.

Several techniques have been proposed to improve the training stability of DEQs. [5] propose to regularize the Jacobian matrix in Eq. (2) during training so that the nonlinear forward system and the backward linear system enjoy appropriate contractivity. [14] propose unrolling-based and



$\mathbf{z}_{\text{out}}$
$\mathbf{u}$
$\mathbf{u}^*$
$\mathbf{u}_0$
$f_\theta(\mathbf{z};\mathbf{x})$
$\mathbf{z}_{\text{out}} = \mathbf{z}$
$\mathbf{u} = \mathbf{u}_{\text{out}}$
$\frac{\partial f_\theta(\mathbf{z}^*;\mathbf{x})}{\partial \mathbf{z}}\mathbf{u} + \frac{\partial L(\mathbf{z}^*,y)}{\partial \mathbf{z}}$
$\mathbf{u}_{\text{out}}$
$\mathbf{z}_0$
$\mathbf{z}^* = \mathbf{z}_N$
$\mathbf{z}$
$L(\mathbf{z}_{N+1}, y)$
$\hat{\mathbf{u}}^*$
$\mathbf{z}_{N+1}$
$\mathbf{u}_{n+1}$
$\mathbf{u}_n$
$\frac{\partial f_\theta(\mathbf{z}_n;\mathbf{x})}{\partial \mathbf{z}}\mathbf{u}_n + \frac{\partial L(\mathbf{z}_n,y)}{\partial \mathbf{z}}$
$\mathbf{z}_n$
$\mathbf{z}_{n+1}$
$L(\hat{\mathbf{z}}_{n+1}, y)$
$\hat{\mathbf{u}}_n$
$\hat{\mathbf{z}}_{n+1}$

(a) Exact gradient (b) Unrolling-based phantom gradient (c) Simultaneous adjoint process (d) Unrolling the intermediates

Figure 1: The gradients proposed for DEQs. (a): the exact gradient [3] solved by an independent fixed-point iteration process. (b): the unrolling-based phantom gradient [14] returned by automatic differentiation on a computational subgraph where the equilibrium state $\mathbf{z}^*$ is unrolled. (c): simultaneous adjoint process along with the forward iterations described in Sec. 4.1. (d): unrolling the intermediate states $\mathbf{z}_n$ for gradient estimation in Sec. 4.2. We leverage (c) and (d) to estimate intermediate gradients and design fully white-box attacks to evaluate the robustness of DEQs.

Neumann-series-based phantom gradients to replace the exact gradient in Eq. (2) for acceleration. The unrolling-based phantom gradient is defined as

$$\lambda \sum_{t=1}^{k} \left(\frac{\partial f_\theta\left(\hat{\mathbf{z}}_{N+t};\mathbf{x}\right)}{\partial(\cdot)} \right) \mathbf{P}_{\lambda,\mathbf{z}_N}^{(t)} \frac{\partial L(\hat{\mathbf{z}}_{N+k}, y)}{\partial \mathbf{z}}, \tag{4}$$

where

$$\mathbf{P}_{\lambda,\mathbf{z}_N}^{(t)} = \prod_{s=t+1}^{k} \left(\lambda \frac{\partial f_\theta\left(\hat{\mathbf{z}}_{N+s};\mathbf{x}\right)}{\partial \mathbf{z}} + (1-\lambda)I \right), \tag{5}$$

$$\hat{\mathbf{z}}_{N+t} = (1-\lambda)\hat{\mathbf{z}}_{N+t-1} + \lambda f_\theta\left(\hat{\mathbf{z}}_{N+t-1};\mathbf{x}\right) \tag{6}$$

are the k unrolling steps with $1 \leq t \leq k$, starting from $\hat{\mathbf{z}}_N = \mathbf{z}^*$ returned by the forward solver.

Eq. (4) is calculated by the automatic differentiation framework [30] on the computational subgraph in Eq. (6). It is demonstrated that the unrolling-based phantom gradient imposes implicit Jacobian regularization effect to DEQ training [14]. DEQs trained by either the exact or the phantom gradients are competitive to deep neural networks in terms of natural accuracy. In our work, we leverage adversarial defense strategies to train DEQs to improve their robustness.

2.2 Adversarial robustness for deep networks

Much research has been dedicated to adversarial attacks and defenses of deep neural networks. On the one hand, white-box adversarial attack techniques like PGD [25] construct adversaries by iteratively perturbing inputs in the gradient ascent direction. The robustness of deep networks is benchmarked by AutoAttack [12], which consists of four attacks including two PGD variants with adaptive stepsize and the query-based SQUARE attack [1]. On the other hand, adversarial training (AT) [25] is one of the most effective defense strategies. By early stopping the training procedure as in [35], the primary PGD-AT framework still achieves competitive robustness performance compared with the state-of-the-art defense techniques like TRADES [43]. It is worth mentioning that many defense approaches claim robustness improvement by obfuscating gradients, which proves to be a false sense of security under adaptive attacks designed specifically [2]. In our work, we train DEQs with PGD-AT and investigate their adversarial robustness by designing customized defenses and adaptive attacks.

3 Challenges for robust general DEQs

This section describes the challenges encountered when we aim to train robust general DEQs.

Misalignment between forward & backward passes. The central idea of DEQs is *directly* solving for the equilibrium state $\mathbf{z}^*$ and differentiating through the fixed point equation $\mathbf{z}^* = f_\theta(\mathbf{z}^*;\mathbf{x})$ for efficient forward and backward passes. Fig. 1-(a) sketches the calculation of the exact gradient [3]. Independent from the forward iterations (the **blue** curve), the exact gradient is acquired by solving for a linear fixed-point system that only depends on the equilibrium state $\mathbf{z}^*$ (the **orange** curve). Fig. 1-(b) shows the calculation of the unrolling-based phantom gradient [14]. $\mathbf{z}^*$ as the final state in the forward pass is unrolled (the **gray** iteration), and the gradient is obtained from the automatic differentiation on the loss function. However, when iterating the gradient computations, the intermediate states $\{\mathbf{z}_n\}$ in the forward pass are bypassed by both methods. The misalignment between the forward and backward tracks results in a gray-box setting for the ready-made attacks.

Convergence of the black-box solvers. In contrast with monotone DEQs, there is no guarantee for the existence and convergence of the equilibrium states in general DEQs. It is thus unknown whether the black-box solvers in DEQs still converge to equilibrium states under input perturbations. Adversarial training also adds the concern on equilibrium convergence. The well-known effect of adversarial training for deep networks is the trade-off between robustness and accuracy [37, 39, 43, 32]. A similar drop in standard accuracy (from 78% to 55%) is also observed for tiny-sized adversarially-trained DEQs [16]. The robustness-accuracy trade-off brings training instability for general DEQs, which may take more iterations in the solvers for equilibrium convergence, or even violate their fixed-point structures. Finally, the robustness comparison is still under-explored between large-sized general DEQs and deep networks with similar parameter counts.

4 On intermediate gradient estimation

As the forward and backward tracks in DEQs are misaligned, the intermediate states in the forward pass are inaccessible to off-the-shelf attacks, which causes gradient obfuscation and results in false positive robustness. Therefore, it is necessary to estimate the intermediate gradients. With the integration of the estimated gradients, the attacks can validate the robustness of DEQs in a fully white-box setting. In this section, we propose two methods for intermediate gradient estimation.

4.1 Simultaneous adjoint in the forward pass

Inspired by the adjoint process in neural ODE models [9], we propose the adjoint process for intermediate gradient estimation in DEQs. The adjoint process in neural ODE models is characterized by an adjoint ODE [31]. For DEQs, we propose to iterate the updates of adjoint states subject to $\mathbf{z}_n$ in the forward pass. We investigate the simultaneous adjoint with Broyden's method [7] as the forward solver. In the forward pass, Broyden's method updates the intermediate state $\mathbf{z}_n$ based on the residual $g_\theta(\mathbf{z}_n;\mathbf{x}) = f_\theta(\mathbf{z}_n;\mathbf{x}) - \mathbf{z}_n$ and B_n, the low-rank approximation of the Jacobian inverse:

$$\mathbf{z}_{n+1} = \mathbf{z}_n - \alpha B_n g_\theta(\mathbf{z}_n;\mathbf{x}),\ \mathbf{z}_0 = \mathbf{0} \tag{7}$$

$$B_{n+1} = B_n + \frac{\Delta\mathbf{z}_{n+1} - B_n \Delta g_{n+1}}{\Delta\mathbf{z}_{n+1}^{\mathrm{T}} B_n \Delta g_{n+1}} \Delta\mathbf{z}_{n+1}^{\mathrm{T}} B_n, \tag{8}$$

where $0 \le n \le N-1$, $B_0 = -I$, $\Delta\mathbf{z}_{n+1} = \mathbf{z}_{n+1} - \mathbf{z}_n$, $\Delta g_{n+1} = g_\theta(\mathbf{z}_{n+1};\mathbf{x}) - g_\theta(\mathbf{z}_n;\mathbf{x})$, and α is the step size. To maintain a simultaneous adjoint, we start from $\mathbf{u}_0 = \mathbf{0}$ and use Broyden's method to solve Eq. (3). Similar with the residual function $g_\theta(\cdot;\mathbf{x})$ for $\mathbf{z}_n$, the fixed-point equation in Eq. (3) defines the residual of the adjoint state. However, we propose to replace the $\mathbf{z}^*$ in Eq. (3) by $\mathbf{z}_n$, and integrate the approximated Jacobian inverse B_n to force the alignment of the adjoint state update:

$$\mathbf{v}_n = \left(\frac{\partial f_\theta(\mathbf{z}_n;\mathbf{x})}{\partial \mathbf{z}}\right)\mathbf{u}_n + \frac{\partial L(\mathbf{z}_n, y)}{\partial \mathbf{z}} - \mathbf{u}_n, \tag{9}$$

$$\mathbf{u}_{n+1} = \mathbf{u}_n - \beta B_n \mathbf{v}_n, \tag{10}$$

where $\mathbf{v}_n$ is the residual at iteration n, $\mathbf{u}_n$ is the updated adjoint state, and $\beta > 0$ is the step size.

We use the following surrogate gradients to construct attacks on the intermediate state $\mathbf{z}_n$:

$$\left[\widetilde{\frac{\partial L}{\partial x}}\right]_n = \left(\frac{\partial f_\theta(\mathbf{z}_n;\mathbf{x})}{\partial \mathbf{x}}\right)\mathbf{u}_n. \tag{11}$$

An illustration for the simultaneous adjoint process is shown in Fig. 1-(c). In the following, we refer to this method as *simultaneous adjoint* when constructing intermediate state attacks in Sec 5.1.

Remark 4.1. We show in Appendix B that under mild assumptions, the $\{\mathbf{u}_n\}$ converges to $\mathbf{u}^*$ when $0 < \beta < 1$ in Eq. (10). However in practice, we do *not* require the convergence of $\mathbf{u}_n$ as we only use them in Eq. (11) as gradient *estimations* to construct *intermediate attacks*.

Remark 4.2. Similar with the update of $\mathbf{u}_n$ in our approach, [16] propose augmented DEQs as an integration of the iterative updating process of $\mathbf{z}$, $\mathbf{u}$, and $\mathbf{x}$ as a whole:

$$F\left(\begin{bmatrix}\mathbf{z}_n \\ \mathbf{u}_n \\ \mathbf{x}_n\end{bmatrix}\right) = \begin{bmatrix} f_\theta(\mathbf{z}_n;\mathbf{x}_n) \\ \left(\frac{\partial f_\theta(\mathbf{z}_n;\mathbf{x}_n)}{\partial \mathbf{z}}\right)\mathbf{u}_n + \frac{\partial L(\mathbf{z}_n,y)}{\partial \mathbf{z}} \\ \mathbf{x}_n - \left(\frac{\partial f_\theta(\mathbf{z}_n;\mathbf{x}_n)}{\partial \mathbf{x}}\right)\mathbf{u}_n \end{bmatrix} \tag{12}$$

The augmented DEQs leverage a black-box solver (e.g., Broyden's method) to find the equilibrium of the whole state $[\mathbf{z}^*, \mathbf{u}^*, \mathbf{x}^*]^{\mathrm{T}}$. However, several cross-terms exist in the joint Jacobian due to the coupling of the three iteration processes, which further hinders the convergence. In contrast, the simultaneous adjoint update in Eq. (10) does not include the update of $\mathbf{x}$. Furthermore, we reuse the Jacobian inverse approximation matrix B_n in the update of $\mathbf{u}_n$, which is easy to implement. Because of the disentanglement of $\mathbf{z}_n$ and $\mathbf{u}_n$ in the joint Jacobian, our method also enjoys better efficiency and flexibility as one can early exit the adjoint process without affecting the updates of $\mathbf{z}_n$'s.

Remark 4.3. Concurrent work [33] also explores the idea of sharing approximated Jacobian inverse B_n in bi-level optimization problems. While their motivation is to accelerate DEQ training, we use the adjoint states to construct gradient estimation and facilitate white-box attacks. We also compare our work with theirs in terms of intermediate state attacks; See Appendix D for details.

4.2 Unrolling the intermediate states

We also propose to estimate the gradient at the state $\mathbf{z}_n$ by unrolling. Depicted in Fig. 1-(d), $\mathbf{z}_n$ is involved in an artificially constructed computational graph. We can thus estimate the intermediate gradient by backpropagation with automatic differentiation. Formally, applying Eq. (4) to $\mathbf{z}_n$ yields

$$\left[\widehat{\frac{\partial L}{\partial \mathbf{x}}}\right]_n^{(k)} = \mathbf{A}_{\lambda,\mathbf{z}_n}^{(k)} \frac{\partial L(\hat{\mathbf{z}}_{n+k}, y)}{\partial \mathbf{z}}, \tag{13}$$

where

$$\mathbf{A}_{\lambda,\mathbf{z}_n}^{(k)} = \lambda \sum_{t=0}^{k-1} \left(\frac{\partial f_\theta(\hat{\mathbf{z}}_{n+t}; \mathbf{x})}{\partial \mathbf{x}} \right) \mathbf{P}_{\lambda,\mathbf{z}_n}^{(k)}, \tag{14}$$

$$\mathbf{P}_{\lambda,\mathbf{z}_n}^{(k)} = \prod_{s=t+1}^{k-1} \left(\lambda \frac{\partial f_\theta(\hat{\mathbf{z}}_{n+s}; \mathbf{x})}{\partial \mathbf{z}} + (1-\lambda) I \right), \tag{15}$$

and the state sequence $\hat{\mathbf{z}}_n, \hat{\mathbf{z}}_{n+1}, \cdots, \hat{\mathbf{z}}_{n+k}$ represents the damped unrolling iteration:

$$\hat{\mathbf{z}}_{n+t} = (1-\lambda)\hat{\mathbf{z}}_{n+t-1} + \lambda f_\theta(\hat{\mathbf{z}}_{n+t-1}; \mathbf{x}), \tag{16}$$

with $t = 1, 2, \cdots, k$ and $\hat{\mathbf{z}}_n = \mathbf{z}_n$. While Eq. (4) is proposed as an approximation of the exact gradient, we unroll the states $\mathbf{z}_n$ for intermediate gradient estimation. Similar to the case of Eq. (11), we use Eq. (13) as estimation to design intermediate attacks for DEQs. We refer to this method as *unrolled intermediates* in the following when incorporating Eq. (13) into the white-box attacks.

5 White-box attacks and defenses for DEQs

This section describes different types of white-box attacks and defense strategies for DEQs.

5.1 White-box attacks for DEQs

The existing attacks leverage the gradients calculated at the *final* state outputted by the forward solver. Based on the surrogate intermediate gradients in Eq. (11) or Eq. (13), we can involve the $\mathbf{z}_n$ in the forward pass into the construction of adversaries. A direct white-box approach is to use the estimated gradient at an *early* state $\mathbf{z}_n$ as an alternative for input perturbations. Another simple yet effective method is to average the intermediate gradients as the gradient ensemble for attacks. For example, the average of all intermediate gradients along the simultaneous adjoint process is given by

$$\sum_n \left[\widetilde{\frac{\partial L}{\partial \mathbf{x}}}\right]_n = \sum_n \left(\frac{\partial f_\theta(\mathbf{z}_n; \mathbf{x})}{\partial \mathbf{x}} \right) \mathbf{u}_n. \tag{17}$$

The gradient ensemble can be viewed as the fusion of all perturbation directions indicated by all $\mathbf{z}_n$'s.

5.2 Defenses with intermediate states

In addition to the final state $\mathbf{z}^*$, the unused intermediate states can be leveraged as well for the defenses of DEQs. A simple yet effective defense strategy is to *early exit* the forward solver during

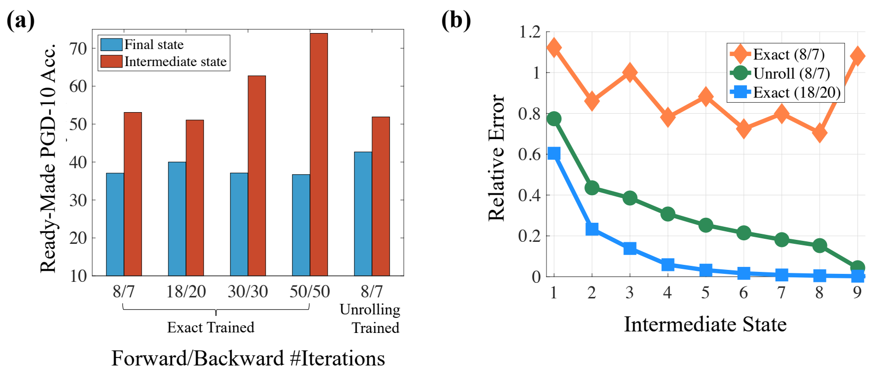


Figure 2: Challenges in benchmarking adversarial robustness of DEQs. (a) Gradient obfuscation issues arise in the DEQs trained with different configurations. With different iteration settings in the DEQ solver or different gradient formulations, the intermediate state *always* exhibit higher robustness than the final state under ready-made PGD-10 attack. (b) Exact-trained DEQ with small iterations violate the fixed-point structure and require more iterations to retain it (analyzed in Sec. 6.1). Both observations motivate us to design adaptive attacks for white-box robustness evaluation for DEQs.

inference. We can evaluate the robustness of DEQs with the early state $\mathbf{z}_n$ in the forward pass, as $\mathbf{z}_n$ and $\mathbf{z}^*$ have the same shape. We determine the optimal timing for early exit by selecting the top robustness performance of all $\mathbf{z}_n$'s on the development set under the ready-made PGD-10 attack.

The input-injected neural network provides an interpretation for the early-state defense. From Eq. (1), the distortion of $\mathbf{z}_n$ comes from both the perturbed $\mathbf{z}_{n-1}$ and the biased transformation $f_\theta(\cdot;\mathbf{x}+\Delta\mathbf{x})$. By early exiting the forward process, one obtains a less distorted intermediate state.

Another defense strategy for DEQs is leveraging the *ensemble* of intermediate states. Similar with Eq. (17), we average the intermediate states $\{\mathbf{z}_n\}$ to defend against attacks. Instead of early stopping, the intermediate state ensemble exploits the state representations at all iterations in the forward solver.

While the proposed defense techniques leverage the intermediate states, they still require only $\mathcal{O}(1)$ memory. For early state defense, we determine the optimal time to early exit the solver on the development set offline for once and then fix the early exit step during testing. For ensemble state defense, we maintain an accumulator to sum up $\{\mathbf{z}_n\}$ along the forward pass without storing them.

6 Experiments

Following the settings in [5], we experiment with the large-sized DEQ with its parameter count similar to ResNet-18 [17]. We also experiment with an XL-sized DEQ with its parameter count similar to WideResNet-34-10 [42] to enable a fair comparison with the empirical robustness of the deep networks. The detailed experimental settings are listed in Appendix A. We first train DEQs on CIFAR-10 [21] with the PGD-AT framework [25], then test the adaptive attacks and defense strategies proposed in Sec. 5 on the adversarially-trained DEQs. We refer to a DEQ as "exact-trained" when using the exact gradient, and "unrolling-trained" when using the unrolling-based phantom gradient in the PGD-AT framework to generate adversaries and optimize for model parameters. Unless specified, all DEQs are adversarially trained in this paper. During training, we use 10-step PGD with the step size of $2/255$ to generate adversaries within the range of $\ell_\infty = 8/255$. For the specific type of attacks, we use PGD and AutoAttack (AA) [13] to instantiate the white-box attacks in Sec. 5.1.

6.1 The retention of the fixed-point structure

We start with the observation on the fixed-point structure. Shown in Fig. 2-(b), the lines illustrate the relative error $\|f_\theta(\mathbf{z}_n;\mathbf{x}) - \mathbf{z}_n\|_2 / \|f_\theta(\mathbf{z}_n;\mathbf{x})\|_2$ for each $\mathbf{z}_n$ [2]. We find that for the exact-trained DEQ with small iteration settings (8 forward / 7 backward iterations), all the relative errors are larger than 0.75, i.e., the forward solver in the DEQ fails to converge to an equilibrium. Such phenomenon

[2]The 9-th intermediate state comes from the implementation in DEQs. In the exact-trained DEQ, $\mathbf{z}_9 = f_\theta(\mathbf{z}_8;\mathbf{x})$. In the unrolling-trained DEQ, $\mathbf{z}_9$ is the final state after unrolling $\mathbf{z}_8$. For the exact-trained DEQ with 18 forward / 20 backward iterations, we only plot the first 9 states for an easy comparison.

Table 2: Performance (%) of the unrolling-trained DEQ-Large with the small (8/7) iteration setting and the exact-trained DEQ-Large with the large (50/50) iteration setting under PGD-10. The "final" rows and columns represent the original DEQ output and the ready-made attacks at the final state. The "early" rows indicate early state defense, and the "intermediate" columns indicate the performance of the strongest intermediate attacks. The rows and the columns of "ensemble" demonstrate the ensemble defense and the white-box attacks based on gradient ensemble. Under the (<u>underlined</u>) strongest attacks, the ensemble defense achieves the best robustness performance (**in bold**).

Training Configurations	Defense	Clean	Simultaneous Adjoint			Unrolled Intermediates		
			Final	Intermediate	Ensemble	Final	Intermediate	Ensemble
(8/7) Unrolling-Trained	Final	78.03	49.81	59.49	54.91	<u>42.67</u>	62.24	51.52
	Early	79.57	54.90	39.19	42.76	51.90	<u>29.38</u>	34.20
	Ensemble	79.67	51.52	52.43	49.47	49.02	55.10	**<u>47.12</u>**
(50/50) Exact-Trained	Final	73.51	37.77	70.52	43.70	<u>36.70</u>	69.29	48.08
	Early	86.98	75.25	<u>12.44</u>	40.12	73.93	18.24	26.22
	Ensemble	75.12	40.20	72.41	45.06	**<u>39.18</u>**	68.83	49.10

reflect the challenge on the convergence of the black-box solvers in DEQs mentioned in Section 3. It is shown that a larger iteration setting is required (18 forward / 20 backward iterations) for exact-trained DEQs to retain the fixed-point structure. In contrast, we find the small iteration setting (8/7) is enough for the unrolling-trained DEQ to retain the fixed-point structure.

Table 1: Performance (%) of the exact-trained DEQ-Large with the small (8/7) solver iterations under different attacks. The high accuracy under PGD-10 with the exact gradient is deteriorated using the unrolling-based phantom gradient. Leveraging the query-based SQUARE leads to even lower accuracy. These observations indicate that the DEQ with violated fixed-point structure suffers severe robustness degradation.

Gradient	Clean	PGD-10	PGD-1000	SQUARE
Exact	78.24	79.97	80.10	5.95
Unrolling		37.07	36.39	

It is necessary to retain the fixed-point structure, otherwise leading to gradient obfuscation issues. As is derived from the implicit differentiation on the fixed-point equation $\mathbf{z}^* = f_\theta(\mathbf{z}^*; \mathbf{x})$, the exact gradient in Eq. (2) becomes inexact when the equilibrium point $\mathbf{z}^*$ is not reached. Table 1 shows the empirical performance of the exact-trained DEQ under the small (8/7) iteration setting. The severe performance degradation under alternative gradient formulations as well as the SQUARE attack also indicates gradient obfuscation, as suggested in [2] and [8].

While large iterations for the exact-trained DEQs keep the fixed-point structure, it inevitably slows down the training speed (detailed in Appendix E.3). For the exact-trained DEQs with the small (8/7) iteration setting, we have also tried with varied Jacobian regularization weights to impose stricter Lipschitz constraints during training, but found the DEQ solver still diverged. We have also analyzed the instability by tracing the variation of Lipschitz constant during the adversarial training of DEQs; See Appendix E.4 for details. By comparison, the unrolling-trained DEQ requires fewer iterations in the forward solver to converge. According to the green line in Fig. 2-(b), the relative errors become lower consequently and reach 0.04 at the final state. The results coincide with [14] that the unrolling-based phantom gradient invokes implicit Jacobian regularization during training.

6.2 Robustness of DEQs under white-box attacks

Intriguingly, we discover the robustness accumulation effect in both the exact-trained and the unrolling-trained DEQs. We plot the highest robustness under the ready-made PGD-10 among all the intermediate states in Fig. 2-(a), with comparison to the final state robustness. It is shown that the intermediate states always exhibit much higher robustness. The accumulated robustness comes from gradient obfuscation, as the ready-made attacks fail to "directly" attack the intermediate states due to misaligned gradients. This resonates with the first challenge in Sec. 3, and similar results are observed as well in adversarially-trained neural ODEs: the large error tolerance from the ODE solvers with adaptive step sizes allows gradient masking after adversarial training [19].

The exact-trained DEQs, as we have discussed in Sec. 6.1, require larger iterations in the solver. However, it is noticed in Fig. 2-(a) that the larger the iteration is in the exact-trained DEQs, the more robust the intermediate states are under ready-made PGD-10. On the contrary, the (8/7) unrolling-trained DEQ still achieves the highest robustness at the final state. To benchmark the white-box robustness, in this section, we compare the (50/50) exact-trained DEQ-Large with the (8/7) unrolling-

Table 3: Performance (%) of the unrolling-trained DEQs under **PGD-10/AutoAttack**. The rows and the columns represent the same meanings as those in Table 2. Under the (underlined) strongest attacks, the ensemble defense achieves the best robustness performance (**in bold**).

Arch.	Defense	Clean	Simultaneous Adjoint (PGD/AA)			Unrolled Intermediates (PGD/AA)		
			Final	Intermediate	Ensemble	Final	Intermediate	Ensemble
Large	Final	78.03	49.81/51.48	59.49/61.95	54.91/52.95	42.67/37.27	62.24/65.53	51.52/49.66
	Early	79.57	54.90/61.12	39.19/55.47	42.76/56.84	51.90/56.86	29.38/25.41	34.20/49.74
	Ensemble	79.67	51.52/56.06	52.43/58.69	49.47/55.02	49.02/50.45	55.10/58.63	**47.12**/**48.37**
XL	Final	82.92	55.80/58.21	55.80/58.21	67.30/64.24	48.58/43.97	65.94/72.76	58.23/69.83
	Early	80.12	51.40/58.92	51.40/58.92	60.78/62.98	52.08/56.58	55.70/62.67	48.88/62.87
	Ensemble	81.17	52.87/58.08	52.87/58.08	61.40/62.23	**51.70**/**54.09**	59.71/66.90	53.23/56.45

trained one [3]. We integrate the estimated intermediate gradients in Sec. 4 as different alternatives in PGD-10 for white-box evaluation. Among all the attack candidates based on intermediate gradients, we select the one that leads to the largest robustness deterioration on the early-state defense in Sec. 5.2 and report the results in Table 2. Ablation studies on the performance of the attack candidates with estimated gradients at different intermediates can be found in Sec. 6.4. We also include the report of memory usage for the defense strategies in Appendix F.1, and the running time complexity analysis for the white-box attacks in Appendix F.2.

Shown in Table 2, for the unrolling-trained DEQ, unrolling the intermediate states results in the strongest attack to the final state and early state defenses. While the robustness accuracies under final and intermediate attacks are improved and better balanced with the ensemble state defense, the ensemble attack leads to the largest performance drop in this case, arriving at the overall white-box robustness of 47.12%. The estimated intermediate gradients based on simultaneous adjoint process also shows significant attack performance for the exact-trained DEQ with large solver iterations. After maximizing the minimum robustness under all attacks across all defense techniques, the overall white-box robustness is 39.18%. In addition, all attacks significantly deteriorate the robustness of the DEQs without adversarial training, indicating that the attacks leveraged in Table 2 are reliably strong (detailed in Sec. 6.5). Considering the superior robustness of the unrolling-trained DEQ as well as its training efficiency, we proceed to experiment with unrolling-trained DEQs for further evaluation.

6.3 Comparison between DEQs and deep networks

In this section, we further provide a thorough evaluation by benchmarking the white-box robustness performance of the unrolling-trained DEQ-Large and DEQ-XL under both PGD-10 and AutoAttack.

Table 3 shows the robustness performance of the unrolling-trained DEQ-Large and DEQ-XL. According to the results, the gradient ensemble attacks are more effective in defeating the early-state defense than the final-state defense. The ensemble attack is the most threatening on the ensemble defense in DEQ-Large. The attack with the gradient at the final state leads to the most substantial performance drop on the ensemble defense in DEQ-XL.

In Table 3, we find that the PGD-10 attack brings more significant performance drops than AutoAttack does in many settings. The results differ from the case in the robustness of deep networks [13]. The phenomenon originates from the difference between intermediate-state attacks and alternative defense strategies. AutoAttack will overfit to the provided gradients at the intermediate or the averaged states, thus generating less threatening adversaries on the defenses based on other states. In addition, the intermediate gradients can also be inaccurate, as they only serve as approximations.

Table 4: The comparison of robustness performance (%) between the DEQs with the ensemble defense and the deep networks of similar sizes. For the DEQs, the weakest robustness under all attacks in Table 3 is reported. For the deep networks, we report the results in [29].

Arch.	Clean	PGD-10	AA	#Params
ResNet-18	82.52	53.58	48.51	10M
DEQ-Large	79.67	47.12	48.37	10M
WRN-34-10	86.07	56.60	52.19	48M
DEQ-XL	81.17	51.70	54.09	48M

The minimum robustness under all types of attacks represents the robustness of a defense strategy. We thus take the most robust defenses for DEQ-Large and DEQ-XL, and compare them with the

[3]The robustness of the (8/7) exact-trained DEQ-Large is much lower than its unrolling-trained counterpart because of the violated fixed-point structure, as shown in Sec. 6.1 and Table 1

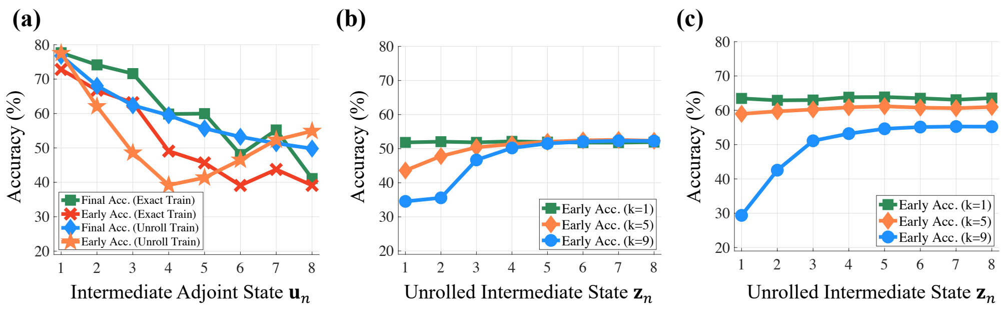


Figure 3: Ablation study on the gradient estimations at different intermediate states. (a) Different robustness performance under PGD-10 with different intermediate adjoint states $\mathbf{u}_n$ as the surrogate gradient. For the approach in Sec. 4.1, $\mathbf{u}_4$ leads to the largest robustness drop in the early state $\mathbf{z}_3$ in the unrolling-based DEQ. (b) and (c): Different unrolled intermediates $\mathbf{z}_n$ with different k's in Eq. (14). the λ in Eq. (13) is set as 0.5 in (b) and 1 in (c). For the method in Sec. 4.2, unrolling the state $\mathbf{z}_1$ with $k = 1$ and $\lambda = 1$ results in the largest robustness drop in the early state $\mathbf{z}_3$.

deep networks of similar parameter counts. Shown in Table 4, the empirical robustness of the DEQs is competitive with or even slightly higher than that of the ResNet-18 and WRN-34-10 models with the PGD-AT framework, respectively.

6.4 Ablation study on different intermediate gradients

In this section, we study the effect of the white-box attacks with gradients estimated at different intermediate states in the forward solver of the large-sized DEQs.

We first inspect the attacks with intermediate gradients acquired from each adjoint state. Fig. 3-(a) plots the robustness of the early-state and final-state defense in both the unrolling-trained and the exact-trained DEQs. For the exact-trained DEQ, due to its violated fixed-point structure, $\mathbf{u}_8$ results in the strongest attack for both the early-state and the final-state defenses. For the unrolling-trained DEQ, the estimated gradients at the consecutive adjoint states $\{\mathbf{u}_n\}$ form increasingly stronger attacks on the robustness of the final state. On the robustness at the early state ($\mathbf{z}_3$), the state $\mathbf{u}_4$ gives rise to the strongest attack, which coincides with Eq. (9) and Eq. (10) that $\mathbf{u}_{n+1}$ directly depends on $\mathbf{z}_n$.

We further explore whether the simultaneous adjoint process are aligned with the forward pass. We use each intermediate adjoint state $\mathbf{u}_n$ as the gradient surrogate in the PGD-10 attack for the unrolling-trained DEQ. Fig. 4 shows the robustness performance of all the intermediate states $\mathbf{z}_n$ in the forward pass of the unrolling-trained DEQ-Large. According to Fig. 4, it always follows that $\mathbf{u}_{n+1}$ results in the largest robustness drop of $\mathbf{z}_n$. As $\mathbf{u}_{n+1}$ directly depends on $\mathbf{z}_n$ in Eq. (10), this validates that the simultaneous adjoint process is aligned with the forward pass at each iteration in terms of adversarial robustness.

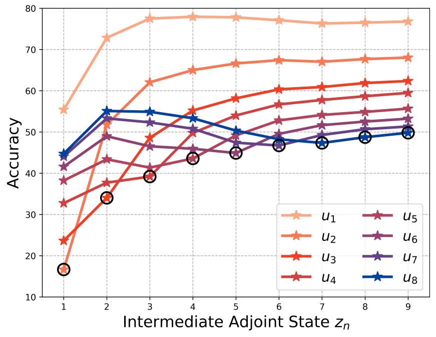


Figure 4: Alignment between the simultaneous adjoint process and the forward pass in the unrolling-trained DEQ.

We also study the effect of unrolling different intermediate states $\{\mathbf{z}_n\}$ for surrogate gradient estimation. Fig. 3-(b)/(c) illustrates the robustness of the early state $\mathbf{z}_3$ in the unrolling-trained DEQ under white-box attacks in different settings. It is shown that the number of unrolling steps for intermediate states like $\mathbf{z}_1$ and $\mathbf{z}_2$ should not be too much in order to obtain a powerful intermediate attack. The reason of this might be the inaccuracy of the unrolled intermediate gradient estimates.

The gradient estimated by unrolling $\mathbf{z}_1$ leads to the most vigorous attack on the robustness of $\mathbf{z}_3$. To understand the circumstance, we note that the unrolling-based intermediate gradient reflects only the feedback from the loss function at the unrolled state in Eq. (13) and Eq. (14). As a result, the estimated gradient may be misaligned with the unrolled state: gradients by unrolling $\mathbf{z}_3$ compose weak attack in terms of the robustness of $\mathbf{z}_3$. It is inferred that the perturbation from the unrolled intermediate gradients must still be propagated in the forward pass to induce enough threatening

Table 5: The performance (%) of the standardly-trained DEQ-Large under ready-made PGD-10.

State	$\mathbf{z}_1$	$\mathbf{z}_2$	$\mathbf{z}_3$	$\mathbf{z}_4$	$\mathbf{z}_5$	$\mathbf{z}_6$	$\mathbf{z}_7$	$\mathbf{z}_8$
Clean Acc.	38.81	82.62	89.63	91.77	92.08	92.29	92.39	92.53
Robust Acc.	2.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 6: Performance (%) of the standardly-trained DEQ-Large [5] with all proposed adaptive attacks and defense strategies under the PGD attack. The notations for the rows and the columns are similar with those in Tables 2 and 3. Under the (<u>underlined</u>) strongest attacks, the ensemble defense still achieves the best robustness performance (**in bold**).

Defense	Clean	Simultaneous Adjoint			Unrolled Intermediates		
		Final	Intermediate	Ensemble	Final	Intermediate	Ensemble
Final	92.53	8.90	11.45	3.69	<u>0.00</u>	<u>0.00</u>	<u>0.00</u>
Early	38.81	6.08	4.54	3.42	2.00	2.94	<u>**1.31**</u>
Ensemble	87.31	9.12	6.39	3.48	<u>0.00</u>	<u>0.00</u>	<u>0.00</u>

distortion. This explains the delay that the unrolled gradient at $\mathbf{z}_1$ affects the robustness of $\mathbf{z}_3$. More ablation studies on the unrolled intermediates can be found in Appendix E.1.

6.5 Performance of the proposed attacks on vanilla DEQ models

In this section, we evaluate the performance of the proposed attacks on the DEQ models without adversarial training. We train a DEQ-Large on CIFAR-10 with standard training following the recipe in [5], and use the ready-made PGD-10 to attack the model. The clean accuracy of each state $\mathbf{z}_n$, as well as its robust accuracy is shown in Table 5. Different from the robustness accumulation effect in the adversarially-trained DEQs (shown in Sec. 6.2, Fig. 2-(a), and Appendix C), the ready-made PGD-10 already has a dramatic effect in attacking all the states in the standardly-trained DEQ.

We proceed to apply all the proposed attacks and defense strategies. Following Sec. 5.2, we determine the optimal timing for early exiting the standardly-trained DEQ as state $\mathbf{z}_1$. Shown in Table 6, it can be seen that all the proposed attacks can defeat the DEQ by standard training. As the white-box robustness of DEQs is assessed by the strongest defense under all attacks (minimum over all columns in a row, then maximum over the minimum of the rows), the white-box robustness of the vanilla DEQ is 1.31% with a 38.81% clean accuracy using the early-state defense. When using the final-state and the ensemble-state defense, the robustness is 0%. These results validate that all the proposed attacks are reliably strong as they all defeat the DEQ models without adversarial training.

7 Conclusion

We study the adversarial robustness of general DEQs, using the exact gradient and the unrolling-based phantom gradient in adversarial training for DEQs, respectively. We observe the gradient obfuscation issues in DEQs under ready-made attacks. Based on the misalignment between the forward and backward tracks, we leverage intermediate states in the forward pass to construct white-box attacks and defense strategies and benchmark the white-box robustness performance of DEQs.

While we have performed a serious comparison of white-box robustness between DEQs and deep networks, it can be seen that the performance of DEQs is on par with that of deep networks. Our empirical observations indicate that we should explore more advanced AT mechanisms for DEQs, in order to exploit their local attractor structures. A potential way is to explicitly encourage closed-loop control during training, similar to the mechanism introduced in [11]. To this end, the gradient estimation method proposed in this paper would be one of the critical ingredients for solving the misalignment between the forward/backward pass of DEQs.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No.61925601) and Beijing Academy of Artificial Intelligence (BAAI). We appreciate all of the anonymous reviewers for their comments and suggestions on this work.

References

[1] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision (ECCV)*, pages 484–501. Springer, 2020.

[2] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning (ICML)*, 2018.

[3] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep Equilibrium Models. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 688–699, 2019.

[4] Shaojie Bai, Vladlen Koltun, and J. Zico Kolter. Multiscale Deep Equilibrium Models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[5] Shaojie Bai, Vladlen Koltun, and J. Zico Kolter. Stabilizing Equilibrium Models by Jacobian Regularization. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 554–565. PMLR, 2021.

[6] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 387–402. Springer, 2013.

[7] C. G. Broyden. A Class of Methods for Solving Nonlinear Simultaneous Equations. *Mathematics of Computation*, 19:577–593, 1965.

[8] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

[9] Ricky T Q Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural Ordinary Differential Equations. In *NeurIPS*, 2018.

[10] Tong Chen, Jean B. Lasserre, Victor Magron, and Edouard Pauwels. Semialgebraic Representation of Monotone Deep Equilibrium Models and Applications to Certification. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[11] Zhuotong Chen, Qianxiao Li, and Zheng Zhang. Towards robust neural networks via close-loop control. In *International Conference on Learning Representations*, 2021.

[12] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. RobustBench: a standardized adversarial robustness benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[13] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning (ICML)*, 2020.

[14] Zhengyang Geng, Xin-Yu Zhang, Shaojie Bai, Yisen Wang, and Zhouchen Lin. On Training Implicit Models. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.

[16] Swaminathan Gurumurthy, Shaojie Bai, Zachary Manchester, and J Zico Kolter. Joint inference and input optimization in equilibrium networks. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[19] Yifei Huang, Yaodong Yu, Hongyang Zhang, Yi Ma, and Yuan Yao. Adversarial Robustness of Stabilized NeuralODEs Might be from Obfuscated Gradients. *CoRR*, abs/2009.13145, 2020.

[20] Saber Jafarpour, Matthew Abate, Alexander Davydov, Francesco Bullo, and Samuel Coogan. Robustness Certificates for Implicit Neural Networks: A Mixed Monotone Contractive Approach. *CoRR*, abs/2112.05310, 2021.

[21] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[22] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations (ICLR)*, 2017.

[23] Guohao Li, Matthias Müller, Bernard Ghanem, and Vladlen Koltun. Training Graph Neural Networks with 1000 Layers. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 6437–6449. PMLR, 2021.

[24] Cheng Lu, Jianfei Chen, Chongxuan Li, Qiuhao Wang, and Jun Zhu. Implicit Normalizing Flows. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[25] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

[26] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, 2016.

[27] Mark Niklas Müller, Robin Staab, Marc Fischer, and Martin T. Vechev. Effective Certification of Monotone Deep Equilibrium Models. *CoRR*, abs/2110.08260, 2021.

[28] Chirag Pabbaraju, Ezra Winston, and J. Zico Kolter. Estimating Lipschitz constants of monotone deep equilibrium models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[29] Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, and Jun Zhu. Bag of tricks for adversarial training. In *International Conference on Learning Representations (ICLR)*, 2021.

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 8024–8035, 2019.

[31] Lev Semenovich Pontryagin, EF Mishchenko, VG Boltyanskii, and RV Gamkrelidze. The mathematical theory of optimal processes, 1962.

[32] Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John Duchi, and Percy Liang. Understanding and mitigating the tradeoff between robustness and accuracy. In *International Conference on Machine Learning (ICML)*, 2020.

[33] Zaccharie Ramzi, Florian Mannel, Shaojie Bai, Jean-Luc Starck, Philippe Ciuciu, and Thomas Moreau. SHINE: SHaring the INverse estimate from the forward pass for bi-level optimization and implicit models. In *International Conference on Learning Representations*, 2022.

[34] Max Revay, Ruigang Wang, and Ian R. Manchester. Lipschitz Bounded Equilibrium Networks. *CoRR*, abs/2010.01732, 2020.

[35] Leslie Rice, Eric Wong, and J Zico Kolter. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning (ICML)*, 2020.

[36] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Representations by Back-propagating Errors. *Nature*, 323(6088):533–536, 1986.

[37] Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. Is robustness the cost of accuracy? – a comprehensive study on the robustness of 18 deep image classification models. In *The European Conference on Computer Vision (ECCV)*, 2018.

[38] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[39] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations (ICLR)*, 2019.

[40] Ezra Winston and J. Zico Kolter. Monotone operator equilibrium networks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[41] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2020.

[42] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *The British Machine Vision Conference (BMVC)*, 2016.

[43] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning (ICML)*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We describe the future work in the Conclusion section. We attach analysis and discuss about the limitations of our work in Appendices.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We discuss them in the Broad Impact section.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Experimental settings

We train DEQs on CIFAR-10 with the PGD-AT framework. The detailed experimental settings for both the unrolling-trained and the exact-trained DEQs are shown in Table 7. The best checkpoint is selected with the top performance on the development set under the ready-made PGD-10 attack. The gradient used in the attack for checkpoint selection is the same as the gradient used for training. The settings on the basics of DEQ training are largely followed from [5]. We leave the integration of the estimated intermediate gradients for adversary generation into AT for DEQs as future work. We use NVIDIA-3090 GPUs for all of our experiments.

Table 7: Detailed hyperparameter settings.

Category	Settings	DEQ-Large	DEQ-XL
Architecture	Input Image Size	32×32	
	Number of Scales	4	
	# of Head Channels for Each Scale	[14, 28, 56, 112]	[20, 40, 80, 160]
	# of Channels for Each Scale	[32, 64, 128, 256]	[72, 144, 288, 576]
	Channel Size of Final Layer	1,680	1,800
	Activation Function	ReLU	
	# of Parameters	10M	48M
DEQ Solver	# of Forward Solver Iterations	8	
	# of Backward Solver Iterations	7	
	Algo. for Forward Solvers	Broyden's Method	
	Algo. for Backward Solvers	Broyden's Method	
Optimization	Optimizer	Adam	
	Learning Rate Schedule	cosine decay	
	Decay Factor	0.1	
	Epochs for Decay	[30, 60, 90]	
	Initial Learning Rate	0.001	
	Nesterov Momentum	0.98	
	Weight Decay	-	
Adv. Training	Batch Size	96	
	Training Epochs	150	
	Pretraining Steps	16,000	
	Weight of JR During Pretraining	-	
	Weight of JR During DEQ Training	0.4	
	Stop Epoch for JR	90	
	Unrolling-Trained: Steps k	5	
	Unrolling-Trained: Damping Factor λ	0.5	
	AT for Pretraining	No	
	Attack in AT for DEQ Training	ready-made PGD-10	
	Grad. for Adv. Generation in AT	same with the gradient used in training	
	Label Smoothing	-	

B On the convergence of simultaneous adjoint

We start with the assumption on nonsingularity of the Jacobian inverse of $g_\theta(\mathbf{z}_n; x) = f_\theta(\mathbf{z}_n; x) - \mathbf{z}_n$:

Assumption B.1. For $\forall \mathbf{z} \in \mathbb{R}^d$ and $\forall \mathbf{x} \in \mathbb{R}^l$, we assume the nonsingularity of Jacobian inverse of $g_\theta(\mathbf{z}; x) = f_\theta(\mathbf{z}; x) - \mathbf{z}$. Namely,

$$\left(\frac{\partial g_\theta(\mathbf{z}; x)}{\partial \mathbf{z}}\right)^{-1} = \left(\frac{\partial f_\theta(\mathbf{z}; \mathbf{x})}{\partial \mathbf{z}} - I\right)^{-1} \tag{18}$$

exists.

We also make assumptions on the precision of B_n, which is the approximation of the Jacobian inverse of $g_\theta(\mathbf{z}_n; x)$.

Assumption B.2. Define

$$\bar{B}_n \left(\frac{\partial g_\theta(\mathbf{z}_n; x)}{\mathbf{z}_n}\right) = I - \boldsymbol{\epsilon}_n(\mathbf{x}). \tag{19}$$

We assume that $\|\boldsymbol{\epsilon}_n(\mathbf{x})\| \leq 1 - \epsilon < 1$ with $\epsilon > 0$ for all $\mathbf{x}$.

With these two assumptions, with $0 < \beta < 1$, we have:

$$\begin{aligned}
&\|\mathbf{u}_{n+1} - \mathbf{u}^*\| && (20)\\
=& \left\|\mathbf{u}_n - \beta \bar{B}_n \mathbf{v}_n - \mathbf{u}^*\right\| && (21)\\
=& \left\|\mathbf{u}_n - \beta \bar{B}_n \left(\left(\frac{\partial f_\theta(\mathbf{z}_n; \mathbf{x})}{\partial \mathbf{z}}\right)\mathbf{u}_n + \frac{\partial L(\mathbf{z}_n, y)}{\partial \mathbf{z}} - \mathbf{u}_n\right) - \mathbf{u}^*\right\| && (22)\\
=& \left\|\mathbf{u}_n - \beta \bar{B}_n \left(\frac{\partial g_\theta(\mathbf{z}_n; x)}{\partial \mathbf{z}}\right)\mathbf{u}_n - \beta \bar{B}_n \frac{\partial L(\mathbf{z}_n, y)}{\partial \mathbf{z}} - \mathbf{u}^*\right\| && (23)\\
=& \left\|\mathbf{u}_n - \beta\left(I - \boldsymbol{\epsilon}_n(\mathbf{x})\right)\mathbf{u}_n - \beta \bar{B}_n \frac{\partial L(\mathbf{z}_n, y)}{\partial \mathbf{z}} - \mathbf{u}^*\right\| && (24)\\
=& \left\|\left((1-\beta)I + \beta \boldsymbol{\epsilon}_n(\mathbf{x})\right)\mathbf{u}_n + \beta\left(I - \boldsymbol{\epsilon}_n(\mathbf{x})\right)\left(I - \frac{\partial f_\theta(\mathbf{z}_n; \mathbf{x})}{\partial \mathbf{z}}\right)^{-1} \frac{\partial L(\mathbf{z}_n, y)}{\partial \mathbf{z}} - \mathbf{u}^*\right\| && (25)\\
=& \left\|\left((1-\beta)I + \beta \boldsymbol{\epsilon}_n(\mathbf{x})\right)(\mathbf{u}_n - \mathbf{u}^*) + \beta\left(I - \boldsymbol{\epsilon}_n(\mathbf{x})\right)\left(\left(I - \frac{\partial f_\theta(\mathbf{z}_n; \mathbf{x})}{\partial \mathbf{z}}\right)^{-1} \frac{\partial L(\mathbf{z}_n, y)}{\partial \mathbf{z}} - \mathbf{u}^*\right)\right\| && (26)\\
\leq& \left\|(1-\beta)I + \beta \boldsymbol{\epsilon}_n(\mathbf{x})\right\| \|\mathbf{u}_n - \mathbf{u}^*\| + \beta \left\|I - \boldsymbol{\epsilon}_n(\mathbf{x})\right\| \left\|\left(I - \frac{\partial f_\theta(\mathbf{z}_n; \mathbf{x})}{\partial \mathbf{z}}\right)^{-1} \frac{\partial L(\mathbf{z}_n, y)}{\partial \mathbf{z}} - \mathbf{u}^*\right\|. && (27)
\end{aligned}$$

Define

$$\mathbf{w}_n = \left(I - \frac{\partial f_\theta(\mathbf{z}_n; \mathbf{x})}{\partial \mathbf{z}}\right)^{-1} \frac{\partial L(\mathbf{z}_n, y)}{\partial \mathbf{z}}. \tag{28}$$

According to the definition of $\mathbf{u}^*$ in Eq. (2), it follows that $\mathbf{w}_n \to \mathbf{u}^*$ when $\mathbf{z}_n \to \mathbf{z}^*$. $\{\mathbf{w}_n\}$ has the highest convergence rate to $\mathbf{u}^*$ as it is the most precised estimation of $\mathbf{u}^*$ at step n. According to Assumption B.2 and $0 < \beta < 1$, the first term in Eq. (27) follows

$$\|(1-\beta)I + \beta \boldsymbol{\epsilon}_n(\mathbf{x})\| \leq (1-\beta) + \beta \|\boldsymbol{\epsilon}_n(\mathbf{x})\| \leq 1 - \beta\epsilon < 1. \tag{29}$$

Substituting Eq. (29) and the upper bound ϵ_0 into Eq. (27), we have

$$\|\mathbf{u}_{n+1} - \mathbf{u}^*\| \leq (1 - \beta\epsilon)\|\mathbf{u}_n - \mathbf{u}^*\| + \|\mathbf{w}_n - \mathbf{u}^*\|, \tag{30}$$

thus

$$\frac{\|\mathbf{u}_{n+1} - \mathbf{u}^*\|}{\|\mathbf{u}_n - \mathbf{u}^*\|} \leq 1 - \beta\epsilon + \frac{\|\mathbf{w}_n - \mathbf{u}^*\|}{\|\mathbf{u}_n - \mathbf{u}^*\|}. \tag{31}$$

We omit the second term in the right-hand side of (31) under the mild assumption of the strictly higher convergence rate of $\{\mathbf{w}_n\}$. As a result, when $n > N$ with N sufficiently large, it follows that $\{\mathbf{u}_n\}$ converges to $\mathbf{u}^*$ as $\|\mathbf{u}_{n+1} - \mathbf{u}^*\| < \|\mathbf{u}_n - \mathbf{u}^*\|$. However, we do *not* require the convergence of $\mathbf{u}_n$ as we only use them in Eq. (11) as gradient *estimations* to construct *intermediate attacks*. In practice, we tune β to facilitate the strongest attacks. For the (8/7) unrolling-trained DEQ-Large/XL, we set $\beta = 0.5$; for the (50/50) exact-trained DEQ-Large, we set $\beta = 0.05$.

C The robustness accumulation effect

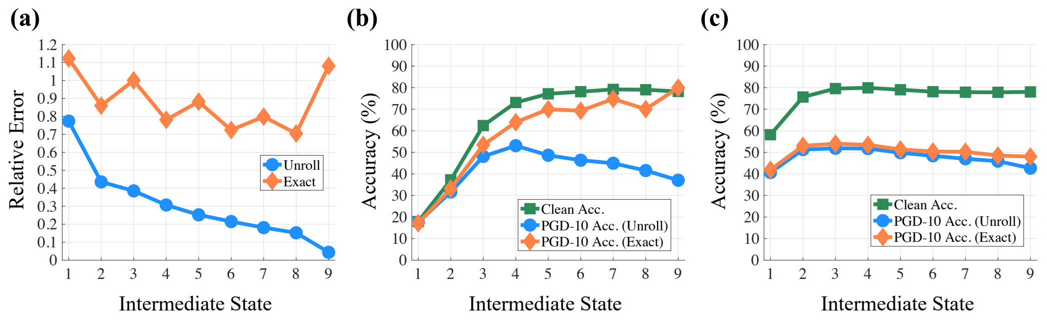


Figure 5: The performance of all intermediate states $\mathbf{z}_n$ in the exact-trained and the unrolling-trained DEQ-Large with 8 forward and 7 backward iterations. (a) Relative error at each intermediate state $\mathbf{z}_n$. The fixed-point structure in the exact-trained DEQ is violated as all relative errors are higher than 1.0. (b) and (c): Robustness evaluated at different intermediate states in (b) the exact-trained and (c) the unrolling-trained DEQs. The unrolling-based phantom gradient forms stronger adversaries in the ready-made PGD-10 attack. The intermediate state $\mathbf{z}_3$ exhibits the strongest robustness in the unrolling-trained DEQ, while $\mathbf{z}_4$ exhibits the strongest robustness in the exact-trained DEQ.

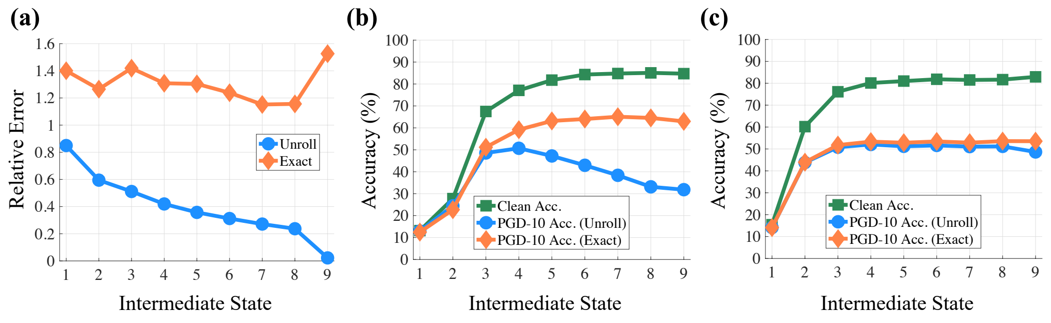


Figure 6: The performance of all intermediate states $\mathbf{z}_n$ in the exact-trained and the unrolling-trained DEQ-XL with 8 forward and 7 backward iterations. (a) Relative error at each intermediate state $\mathbf{z}_n$. The fixed-point structure in the exact-trained DEQ is violated as all relative errors are higher than 1.0. (b) and (c): Robustness evaluated at different intermediate states in (b) the exact-trained and (c) the unrolling-trained DEQs. The unrolling-based phantom gradient forms stronger adversaries in the ready-made PGD-10 attack. For both DEQs, the intermediate state $\mathbf{z}_4$ exhibits stronger robustness than other states.

Figures 5 and 6 the robustness accumulation effect in the DEQs with 8 forward and 7 backward iterations. For both the exact-trained and the unrolling-trained DEQs, the robustness accumulation effect exists because the black-box solvers have obfuscated the gradients used in the ready-made attacks. It can also be seen that the fixed-point structures are broken in the exact-trained DEQ-Large and DEQ-XL under this setting. Figure 2 show that the exact-trained DEQs with more iterations in the solvers can retain the fixed-point structure (more details in Appendix E.3).

D Comparison with the adjoint Broyden method

Similar to our simultaneous adjoint design, concurrent work [33] propose the adjoint Broyden method to share the approximated Jacobian inverse B_n into the backward pass. Their motivation, however, is to accelerate DEQ training while we integrate the simultaneous adjoint into ready-made attacks to facilitate white-box robustness evaluation. In this section, we compare the adjoint Broyden method with our simultaneous adjoint in terms of the effect of intermediate/ensemble attacks.

Table 8: Comparison between the proposed simultaneous adjoint and the adjoint Broyden method.

Arch.	Defense	Clean	PGD : Simultaneous Adjoint / Adjoint Broyden		
			Final	Intermediate	Ensemble
DEQ-Large	Final	78.03	**49.81**/52.34	59.49/**58.22**	**54.91**/58.06
	Early	79.57	54.90/**45.29**	**39.19**/43.48	**42.76**/45.47
	Ensemble	79.67	51.52/**49.24**	52.43/**49.24**	**49.47**/53.04
DEQ-XL	Final	82.92	**55.80**/65.00	**55.80**/65.00	**67.30**/71.20
	Early	80.12	**51.40**/59.26	**51.40**/59.26	**60.78**/65.67
	Ensemble	81.17	**52.87**/60.22	**52.87**/60.22	**61.40**/66.69

Table 8 shows the comparison between our simultaneous adjoint and the adjoint Broyden method with PGD as the attack. The proposed simultaneous adjoint results in stronger white-box attacks in general. The adjoint Broyden method also yields better performance in some cases. We further compare among the proposed simultaneous adjoint, the adjoint Broyden method, and the unrolled intermediates in Table 9.

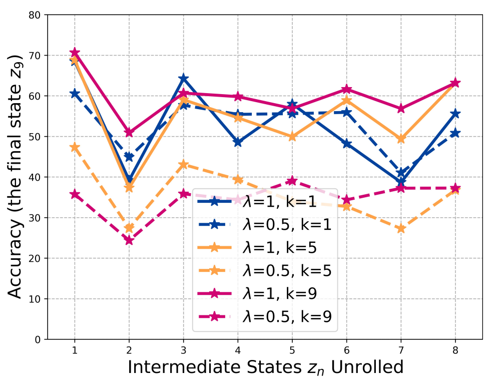
Table 9: Comparison among the proposed simultaneous adjoint, the adjoint Broyden method, and the (proposed) unrolled intermediates with PGD.

Arch.	Defense	Clean	PGD : Simultaneous Adjoint / Adjoint Broyden / Unrolling Intermediates		
			Final	Intermediate	Ensemble
DEQ-Large	Final	78.03	49.81/52.34/**42.67**	59.49/**58.22**/62.24	54.91/58.06/**51.52**
	Early	79.57	54.90/**45.29**/51.90	39.19/43.48/**29.38**	42.76/45.47/**34.20**
	Ensemble	79.67	51.52/49.24/**49.02**	52.43/**49.24**/55.10	49.47/53.04/**47.12**
DEQ-XL	Final	82.92	55.80/65.00/**48.58**	**55.80**/65.00/65.94	67.30/71.20/**58.23**
	Early	80.12	**51.40**/59.26/52.08	**51.40**/59.26/55.70	60.78/65.67/**48.88**
	Ensemble	81.17	52.87/60.22/**51.70**	**52.87**/60.22/59.71	61.40/66.69/**56.45**

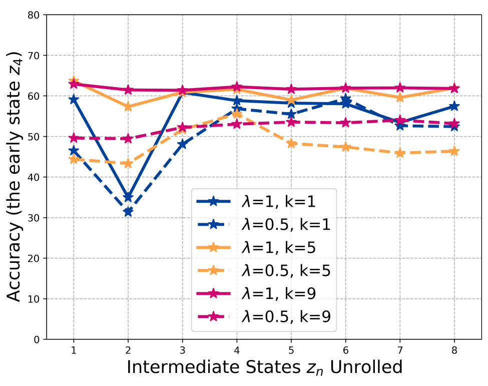
Table 9 shows that the unrolled intermediates still have the dominant effect in facilitating white-box attacks in general. In the future work, we will integrate our simultaneous adjoint/the adjoint Broyden method into the adversarial training process of DEQs and validate their white-box robustness.

E Ablation studies

E.1 Settings of the unrolled intermediates

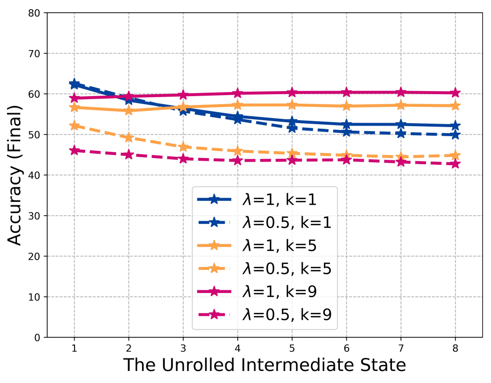


(a) Exact-trained, defense with the final state

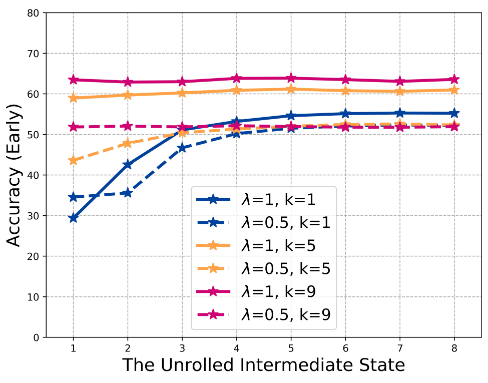


(b) Exact-trained, defense with the early state

Figure 7: Ablation on unrolling different intermediate states with different steps in exact-trained DEQ-Large.



(a) Unrolling-trained, defense with the final state



(b) Unrolling-trained, defense with the final state

Figure 8: Ablation on unrolling different intermediate states with different steps in unrolling-trained DEQ-Large.

E.2 The backward solver iteration threshold in attacks with exact gradients

The exact gradient in Eq. (2) is solved by an independent fixed-point iteration process solely based on $\mathbf{z}^*$. While this iteration process is not aligned with the forward pass, one might still question whether the intermediate states in the backward solver can lead to strong attacks.

We address the question by experimenting with the unrolling-trained DEQ-Large. We use the exact gradient solved by a backward solver for adversary generation in PGD-10. While the default iteration threshold is 7, we investigate the capability of all the 7 intermediate states in constructing adversaries. We also increase the number of the iterations up to 20 to see whether the robustness is degraded. The results are shown in Fig. 9.

From Fig. 9, it can be seen that the gradients from less iterations result in less powerful attacks for both the final and the early states. With increased backward iterations, the robustness of both states is dropped by about 1%. However, as shown in Table 2, PGD-10 with the unrolling-based phantom gradient results in the robustness of 42.67 for the final state and 51.90 for the early state, both of which lower than those in Fig. 9. It is thus concluded that alternating the backward iterations in the exact gradient does not have significant effect on deteriorating the robustness of the unrolling-trained DEQ.

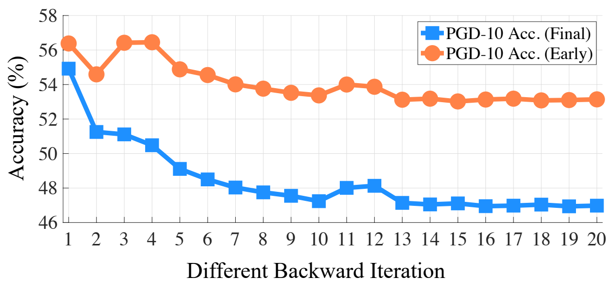


Figure 9: Robustness of the final state as well as the early state in the unrolling-trained DEQ-Large under PGD-10. The gradients in the attacks are returned from the backward solver with different iterations.

E.3 Solver iteration thresholds in the exact-trained DEQs

We increase the iteration thresholds of the forward and the backward solvers in the exact-trained DEQs to stabilize adversarial training. We experiment with the exact-trained DEQ-Large and report the results in Table 10 (which is also illustrated in Fig. 2-(a)).

Table 10: Exact-trained DEQ-Large with larger iteration thresholds in the forward and the backward solvers during training. The "PGD, Unroll" column, demonstrating the accuracy of the final and the top intermediate state under ready-made PGD attacks with the unrolling-based phantom gradient, is used to illustrate Fig. 2-(a).

Training	(ForIter,/BackIter)	Defense	Clean	PGD, Exact	PGD, Unroll	Speed (Samples/s)	Note
Exact	(8/7)	Final	78.24	79.97	37.07	28.3	Grad. Obfus.
		Early	73.12	63.95	53.09		
	(18/20)	Final	73.16	39.92	40.00	22.3	-
		Early	72.02	51.44	51.07		
	(30/30)	Final	81.17	41.51	37.12	7.6	-
		Early	87.72	65.22	62.74		
	(50/50)	Final	73.51	37.29	36.70	5.3	-
		Early	86.98	74.04	73.93		
Unrolling	(8/7)	Final	78.03	48.03	42.67	**35.4**	-
		Early	79.57	54.01	51.90		

From Table 10, it is witnessed that increasing the forward and the backward iterations imposes stabilization effect on DEQ training with the exact gradient. As a consequence, no false-positive robustness is observed in the settings of iteration pairs $(18/20)$. The relative error for each state $\mathbf{z}_n$ in the forward pass under the $(18/20)$ setting is shown in Fig. 10-(a).

In this setting, the forward solver may require less number of iterations than the threshold 18. Here we plot the relative errors of the first 11 states in the forward solver. From Fig. 10-(a) (also the blue line in Fig. 2-(b)), the $(18/20)$ exact-trained DEQ does not violate its fixed-point structure, thus the gradient obfuscation issue is avoided. However, both of its training speed and robustness performance are outperformed by the unrolling-trained DEQ in the $(8/7)$ setting.

Similar to Figures 5 and 6, we also compare the robustness accumulation effect between the $(18/20)$ exact-trained DEQ with the $(8/7)$ unrolling-trained DEQ. Figure 10-(b) shows the comparison: clean and PGD-10 accuracies of both the final and the top intermediate states in the (18/20) exact-trained DEQ are lower than the (8/7) unrolling-trained DEQ. This is similar with the conclusion drawn from Table 2.

E.4 Jacobian regularization weights during training

Another way to stabilize the adversarial training process is to impose stricter regularization on the DEQs. [5] propose Jacobian regularization to stabilize the standard training of DEQs. In this section, we vary the weight γ of the Jacobian regularization during DEQ training to its effect. In standard

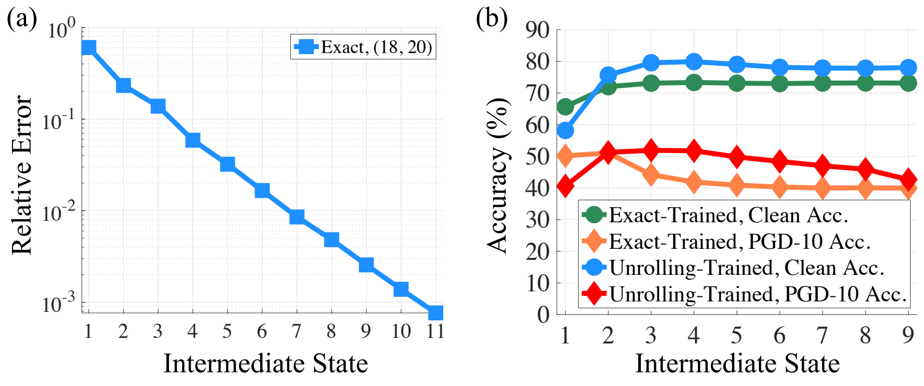


Figure 10: Performance of the (18/20) exact-trained DEQ-Large. (a) The relative error of each state $\mathbf{z}_n$ in the forward pass of the exact-trained DEQ-Large under the $(18/20)$ setting. (b) Robustness accumulation effect in DEQ-Large: the (18/20) exact-trained DEQ v.s. the (8/7) exact-trained DEQ.

training, the weight is set as 0.4. In this section, we sweep γ over $\{0.4, 0.8, 1.2, 1.6, 2.0, 3.2\}$ to train DEQ-Large with the exact gradient with 8 forward and 7 backward iterations (8/7).

Similar to Fig. 2-(b), we plot the relative error at each $\mathbf{z}_n$'s under different γ's settings.

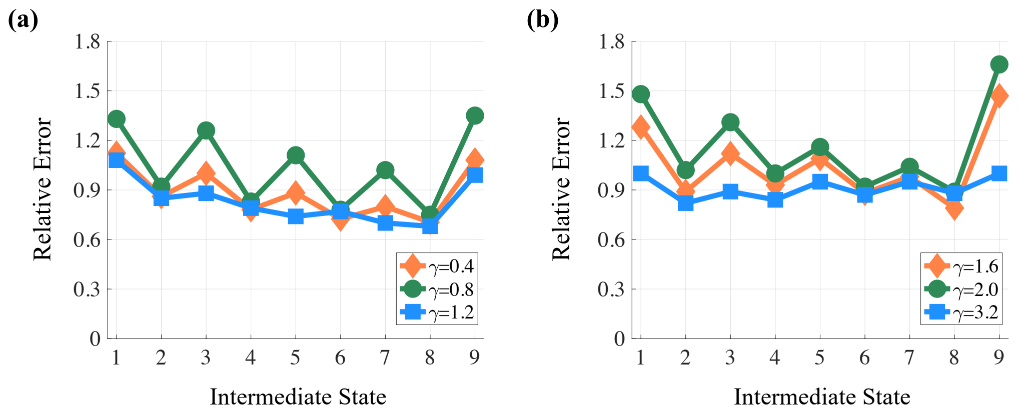


Figure 11: Relative errors at each intermediate state $\mathbf{z}_n$ under different γ's settings.

Illustrated in Fig. 11, all of the relative errors are larger than 0.6, indicating the violation of the fixed-point structure. We have also explored the stability of the adversarial training process in the $\gamma = 3.2$, (8/7) exact-trained setting and the $\gamma = 0.4$, (8/7) unrolling-trained setting. We calculate the averaged spectral radius on the development set for all the checkpoints along the training, and plot them, together with their accuracy under the ready-made PGD-10 attack, in Figs 12 and 13.

Shown in Fig. 13, the $\gamma = 0.4$, (8/7) unrolling-trained DEQ-Large model always has a spectral radius less than or around 1.0. In contrast, the spectral radius of the $\gamma = 3.2$, (8/7) exact-trained DEQ-Large model becomes far larger than 1.0 since the 65-th checkpoint in Fig. 12. This coincides with the violated fixed-point structure, although achieving high (> 60, but false-positive) accuracy under the ready-made PGD-10 attack. In this work, we use the checkpoint with the highest robustness under the ready-made PGD10 along the adversarial training process for our study. We leave the study of the white-box robustness evaluation for other checkpoints in future work.

When the fixed-point structure is broken, query-based attacks have a drastic effect on reducing the classification accuracy of DEQs (see Table 1). We use SQUARE to attack these exact-trained DEQ-Large models with varied Jacobian regularization weight γ's.

Table 11 compares the performance of the models with different γ's under SQUARE and the ready-made PGD attack. For the (8/7) exact-trained DEQ-Large with varied γ's, the SQUARE attack

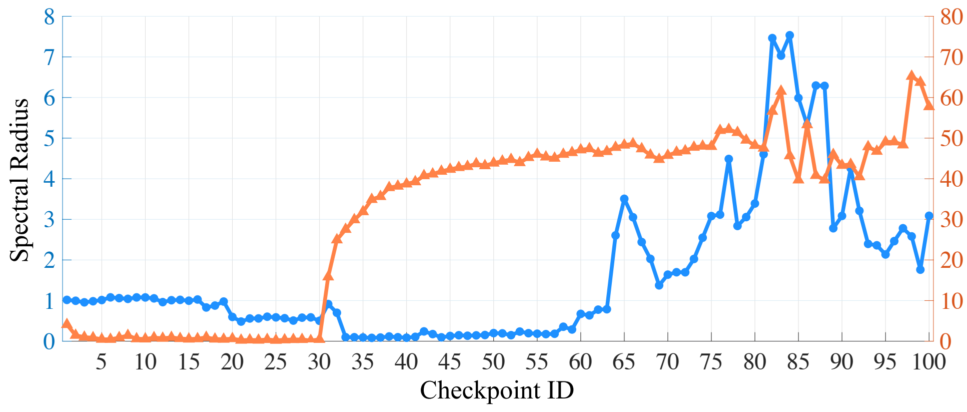


Figure 12: $\gamma = 3.2$, (8/7) exact-trained DEQ-Large. The trace of spectral radius on the development set and the accuracy of each checkpoint under the ready-made PGD-10 attack. The blue line traces the spectral radius, and the orange line traces the accuracy under ready-made PGD-10.

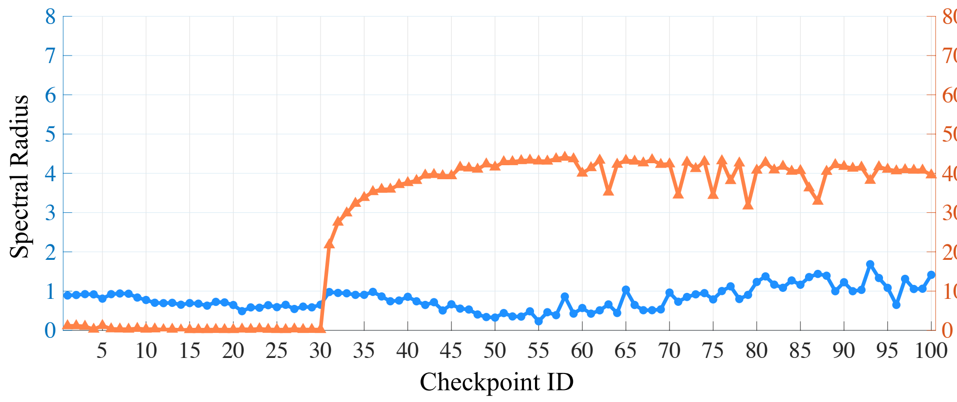


Figure 13: $\gamma = 0.4$, (8/7) unrolling-trained DEQ-Large. The trace of spectral radius on the development set and the accuracy of each checkpoint under the ready-made PGD-10 attack. The blue line traces the spectral radius, and the orange line traces the accuracy under ready-made PGD-10.

appears to be more powerful than the ready-made (gradient-based) PGD attack and always leads to severe robustness degradation. According to [2, 8], such a phenomenon indicates gradient obfuscation. For the exact-trained DEQ-Large with more iterations in the DEQ solver and the unrolling-trained DEQ-Large, as they have retained the fixed-point structure (Fig. 2-(b) and Fig. 10-(a)), we find the gradient-based PGD attack to be more effective. However, we emphasize that the two models *still* suffer from gradient obfuscation: shown in Fig. 10-(b), robustness accumulation effect is observed in both of the models (despite the retained fixed-point structure). The reason of the effect, as we have mentioned in Sec. 6.2, is that the black-box solver in DEQs results in misaligned gradients, which avoids the ready-made attacks to "directly" attack the intermediate states. This has motivated us to propose intermediate/ensemble attacks and defenses for white-box robustness evaluation.

To summarize, we have studied two types of gradient obfuscation in our work (see Table 12). The violated fixed-point structure can be remedied by different techniques: we adopt unrolling-trained DEQs in Sec. 6.1, increase the solver iterations in Sec. 6.2 and Appendix E.3, and attempt with stricter regularization in Appendix E.4. We will explore with more regularization techniques in future work. However, the robustness accumulation effect *always* exists. This ultimately urges the necessity of white-box robustness evaluation, and we propose several white-box attacks and defenses in this work. The two types of gradient obfuscation also echo with the two challenges in Sec. 3.

Table 11: (8/7) exact-trained DEQ-Large with varied Jacobian regularization weight γ's, in comparison with ($\gamma = 0.4$): the (18/20) exact-trained DEQ-Large and the (8/7) unrolling-trained DEQ-Large.

γ	0.4	0.8	1.2	1.6	2.0	3.2	0.4 (18/20)	0.4 (Unroll)
Clean	78.24	82.27	80.64	78.82	67.34	66.70	73.16	78.03
PGD (*ready-made*)	79.97	71.00	48.71	62.23	61.77	65.22	39.92	42.67
SQUARE	5.95	10.00	32.54	23.85	4.21	2.72	47.20	45.34

Table 12: To summarize our work: the two types of gradient obfuscation that we have studied, and what we have done for our white-box robustness evaluation.

State investigated	Phenomenon observed	The reason of the phenomenon	What we have done
Final	SQUARE is more effective	Violated fixed-point structure	"Fix" the structure by: Using the unrolling-trained DEQs; Increasing the iterations in the solver; Attempting with varied regularization weights.
Intermediate	Robustness accumulation	Black-box solvers	Propose several white-box attacks and defenses.

F Memory and time complexity

F.1 On the O(1) memory concern of the proposed attacks and defenses

The unique property of DEQs lies in its O(1) memory consumption. It is therefore necessary to study whether the O(1) memory constraint still applies in the proposed attacks and defenses.

Table 13: The memory usage of different defense strategies used in the (8/7) unrolling-trained DEQ-Large. No extra computation is needed.

Defense	Mem (GB)
Final	3.77
Early	3.77
Ensemble	3.77

From the attacking aspect, white-box attackers are assumed to have full access to the model, therefore they are allowed to "open the black box" to trace all the intermediate steps in the solver. To fully evaluate the worst-case performance of models, white-box attackers are usually not constrained by the O(1) memory. From the defending aspect, as stated in Sec. 5.2, our defense methods still require only O(1) memory. For early state defense, we determine the optimal time to early exit the solver on the development set offline for once and then fix the early exit step during testing. For ensemble state defense, we maintain an accumulator ($\mathrm{ret} \mathrel{+}= \mathbf{z}_n$) without storing the intermediate states of $\mathbf{z}_n$ and output with ret/N instead of $\mathbf{z}_N$: Table 13 shows the empirical results on memory usage.

F.2 On the time complexity concern of simultaneous adjoint

The core idea of the simultaneous adjoint is to **reuse** the approximated Jacobian inverse B_n in the forward calculation of $\mathbf{z}_n$ when calculating the adjoint state $\mathbf{u}_n$. As a result, the "approximated Jacobian inverse" in the simultaneous adjoint does not need extra calculation. This is different from the original DEQ design where the forward and the backward passes are decoupled by separate fixed-point solvers, where different B_n's need to be maintained separately.

Specifically, compared with the original forward pass (Eqs. (7) and (8)), the simultaneous adjoint calculation augments it with Eqs. (9) and (10). The time complexity of Eq.(10) is equilavent to Eq.(7). Eq. (9) calculates the residual of Eq.(3), the fixed-point equation of the backward pass, at $\mathbf{u}^* = \mathbf{u}_n$ and $\mathbf{z}^* = \mathbf{z}_n$. The complexity of this calculation is equivalent to just the **residual evaluation** when solving for the exact gradient in Eq.(3).

In practice, the running time for the adaptive PGD-10 attack with the final adjoint state in "Simultaneous Adjoint" is 6,479ms per batch. In comparison, the running time for the adaptive PGD-10 attack with the unrolled final state in "Unrolled Intermediates" is 5,369ms per batch. It can be seen that the introduced computational burden of Eqs.(9) and (10) does not take the majority of the running time.

G Limitations and Broader Impact

Adversarial attacks are fatal to deep learning methods, causing severe security risks in model deployment. To this end, reliable techniques are needed to defend against the attacks. In this work, we study the white-box robustness of general DEQ models. We observe the gradient obfuscation effect with ready-made attacks and propose several white-box attacks and defenses to facilitate white-box robustness evaluation. Our work contributes to the safety of general DEQs in white-box settings.

In this work, we did not test our methods on the adversarially-trained DEQs on ImageNet due to the time limit. Recent work [4, 5] has shown that DEQ models work well on large-scale vision tasks, including ImageNet and Cityscapes. Given this, we could apply advanced adversarial training algorithms [41] to DEQs, which preserves the scalability of our methods. We also leave the white-box robustness evaluations on ImageNet as our future work. In addition, future work also includes efficient stabilization of the adversarial training procedure, as well as advanced white-box attacks and defense strategies for DEQs.