Continuous Deep Equilibrium Models: Training Neural ODEs Faster by Integrating Them to Infinity

Avik Pal
Electrical Engg. & Computer Science
MIT
Cambridge, U.S.A.
avikpal@mit.edu

Alan Edelman
Mathematics
MIT
Cambridge, U.S.A.
edelman@mit.edu

Chris Rackauckas
Mathematics
MIT
Cambridge, U.S.A.
crackauc@mit.edu

***Abstract*—Implicit models separate the definition of a layer from the description of its solution process. While implicit layers allow features such as depth to adapt automatically to new scenarios and inputs, this adaptivity makes its computational expense challenging to predict. In this manuscript, we *increase the "implicitness" of the DEQ by redefining the method in terms of an infinite time neural ODE*, which paradoxically decreases the training cost over a standard neural ODE by $2-4\times$. Additionally, we address the question: *is there a way to simultaneously achieve the robustness of implicit layers while allowing the reduced computational expense of an explicit layer?* To solve this, we develop Skip and Skip Reg. DEQ, an implicit-explicit (IMEX) layer that simultaneously trains an explicit prediction followed by an implicit correction. We show that training this explicit predictor is free and even decreases the training time by $1.11-3.19\times$. Together, this manuscript shows how bridging the dichotomy of implicit and explicit deep learning can combine the advantages of both techniques.**

***Index Terms*—implicit neural networks, neural ode, deep equilibrium models, steady-state problems**

I. Introduction

Implicit layer methods, such as Neural ODEs and Deep Equilibrium models [1, 2, 3], have gained popularity due to their ability to automatically adapt model depth based on the "complexity" of new problems and inputs. The forward pass of these methods involves solving steady-state problems, convex optimization problems, differential equations, etc., all defined by neural networks, which can be expensive. However, training these more generalized models has empirically been shown to take significantly more time than traditional explicit models such as recurrent neural networks and transformers. *Nothing within the problem's structure requires expensive training methods, so we asked, can we reformulate continuous implicit models so that this is not the case*?

[4, 5, 6, 7] have identified several problems with training implicit networks. These models grow in complexity as training progresses, and a single forward pass can take over 100 iterations [6] even for simple problems like MNIST. Deep Equilibrium Models [2, 8] have better scaling in the backward pass but are still bottlenecked by slow steady-state convergence. [9] quantified several convergence and stability problems with DEQs. They proposed a regularization technique by exploiting the "implicitness" of DEQs to stabilize their training. *We marry the idea of faster backward pass for DEQs and continuous modeling from Neural ODEs to create Infinite Time Neural ODEs, which scale significantly better in the backward pass and drastically reduce the training time.*

Our main contributions include[1]

1) An improved DEQ architecture (Skip-DEQ) that uses an additional neural network to predict better initial conditions.
2) A regularization scheme (Skip Regularized DEQ) incentivizes the DEQ to learn simpler dynamics and leads to faster training and prediction. Notably, this does not require nested automatic differentiation and thus is considerably less computationally expensive than other published techniques.
3) A continuous formulation for DEQs as an infinite time neural ODE, which paradoxically accelerates the backward pass over standard neural ODEs by replacing the continuous adjoints with a simple linear system.
4) We demonstrate the seamless combination of Continuous DEQs with Skip DEQs to create a drop-in replacement for Neural ODEs without incurring a high training cost.

II. Background

Explicit Deep Learning Architectures specify a projection $f : \mathcal{X} \mapsto \mathcal{Z}$ by stacking multiple "layers". Implicit models, however, define a solution process instead of directly specifying the projection. These methods enforce a constraint on the output space $\mathcal{Z}$ by learning $g : \mathcal{X} \times \mathcal{Z} \mapsto \mathbb{R}^n$. By specifying a solution process, implicit models can effectively vary features like depth to adapt automatically to new scenarios and inputs. Some prominent implicit models include Neural ODEs [1], where the output z is defined by the ODE $\frac{dz}{dt} = g_\phi(x, t)$. [10] generalized this framework to Stochastic Differential Equations (SDEs) by stochastic noise injection, which regularizes the training of Neural ODEs, allowing them to be more robust and achieve better generalization. [2] designed equilibrium models where the output z was constrained to be a steady state, $z^* = f_\theta(z^*, x)$. Another example of implicit layer architectures is seen in [11, 12] set z to be the solution of convex optimization problems.

Deep Implicit Models essentially removed the design bottleneck of choosing the "depth" of neural networks. Instead, these

[1] Code: https://github.com/SciML/DeepEquilibriumNetworks.jl

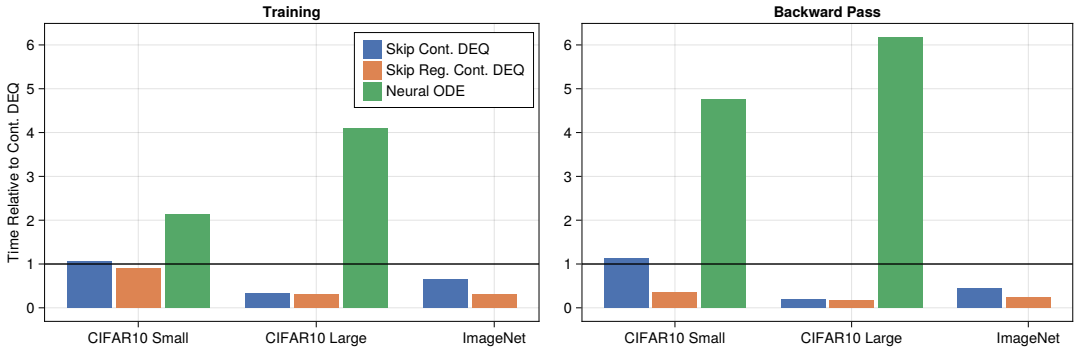

Fig. 1: **Relative Training and Backward Pass Timings against Continuous DEQs** *(lower is better)*: In all scenarios, Neural ODEs take $4.7 - 6.182\times$ more time in the backward pass compared to Vanilla Continuous DEQs. Whereas combining Skip (Reg.) with Continuous DEQs accelerates the backward pass by $2.8 - 5.9\times$.

models use a "tolerance" to determine the accuracy to which the constraint needs to be satisfied. Additionally, many of these models only require $O(1)$ memory for backpropagation, thus alluding to potential increased efficiency over their explicit layer counterparts. However, evaluating these models require solving differential equations [1, 10], non-linear equations [2], convex optimization problems [11, 12], etc. Numerous authors [4, 5, 6, 7, 9, 13] have noted that this solution process makes implicit models significantly slower in practice during training and prediction compared to explicit networks achieving similar accuracy.

A. Neural Ordinary Differential Equations

Initial Value Problems (IVPs) are a class of ODEs that involve finding the state at a later time t_1, given the value z_0 at time t_0. [1] proposed the Neural ODE framework, which uses neural networks to model the ODE dynamics

$$\frac{dz(t)}{dt} = f_\theta\left(z\right)$$

Using adaptive time stepping allows the model to operate at a variable continuous depth depending on the inputs. Removing the fixed depth constraint of Residual Networks provides a more expressive framework and offers several advantages in problems like density estimation [4], irregularly spaced time series problems [14], etc. Training Neural ODEs using continuous adjoints has the added benefit of constant memory overhead. However, this benefit often leads to slower training since we need to backsolve an ODE. We defer the exact details of the continuous adjoint equations to [1].

B. Deep Equilibrium Models

Deep Equilibrium Networks (DEQs) [2] are implicit models where the output space represents a steady-state solution. Intuitively, this represents infinitely deep neural networks with input injection, i.e., an infinite composition of explicit layers $z_{n+1} = f_\theta(z_n, x)$ with $z_0 = 0$ and $n \to \infty$. In practice, it is equivalent to evaluating a dynamical system until it reaches a steady state $z^* = f_\theta(z^*, x)$. [2, 8] perform nonlinear fixed point iterations of the discrete dynamical system using Broyden's method [8, 15] to reach this steady-state solution.

Evaluating DEQs requires solving a steady-state equation involving multiple evaluations of the explicit layer slowing down the forward pass. However, driving the solution to steady-state makes the backward pass very efficient [16]. Despite a potentially infinite number of evaluations of f_θ in the forward pass, backpropagation only requires solving a linear equation.

$$\begin{aligned} & z^* = f_\theta(z^*, x) \\ \implies & \frac{\partial z^*}{\partial \theta} = \frac{f_\theta(z^*, x)}{\partial z^*} \cdot \frac{\partial z^*}{\partial \theta} + \frac{\partial f_\theta(z^*, x)}{\partial \theta} \\ \implies & \left(I - \frac{\partial f_\theta(z^*, x)}{\partial z^*}\right) \frac{\partial z^*}{\partial \theta} = \frac{\partial f_\theta(z^*, x)}{\partial \theta} \end{aligned}$$

For backpropagation, we need the Vector-Jacobian Product (VJP):

$$\begin{aligned} \left(\frac{\partial z^*}{\partial \theta}\right)^T v &= \left(\frac{\partial f_\theta(z^*, x)}{\partial \theta}\right)^T \left(I - \frac{\partial f_\theta(z^*, x)}{\partial z^*}\right)^{-T} v \\ &= \left(\frac{\partial f_\theta(z^*, x)}{\partial \theta}\right)^T g \end{aligned}$$

where v is the gradients from layers after the DEQ module. Computing $\left(I - \frac{\partial f_\theta(z^*, x)}{\partial z^*}\right)^{-T}$ is expensive and makes DEQs non-scalable to high-dimensional problems. Instead, we solve the linear equation $g = \left(\frac{\partial f_\theta(z^*, x)}{\partial z^*}\right)^T g + v$ using Newton-Krylov Methods like GMRES [17]. To compute the final VJP, we need to compute $\left(\frac{\partial f_\theta(z^*, x)}{\partial \theta}\right)^T g$, which allows us to efficiently perform the backpropagation without explicitly computing the Jacobian.

1) Multiscale Deep Equilibrium Network: Multiscale modeling [18] has been the central theme for several deep computer vision applications [19, 20, 21, 22]. The standard DEQ formulation drives a single feature vector to a steady state. [8] proposed Multiscale DEQ (MDEQ) to learn coarse and fine-grained feature representations simultaneously. MDEQs operate at multiple feature scales $z = \{z_1, z_2, \ldots, z_n\}$, with the new equilibrium state $z^* = f_\theta(z_1^*, z_2^*, \ldots, z_n^*, x)$. All the feature vectors in an MDEQ are interdependent and are simultaneously driven to a steady state. [8] used a Limited-Memory Broyden Solver [15] to solve these large scale computer vision problems.

We use this MDEQ formulation for all our classification experiments.

2) Jacobian Stabilization: Infinite composition of a function f_θ does not necessarily lead to a steady-state – chaos, periodicity, divergence, etc., are other possible asymptotic behaviors. The Jacobian Matrix $J_{f_\theta}(z^*)$ controls the existence of a stable steady-state and influences the convergence of DEQs in the forward and backward passes. [9] describes how controlling the spectral radius of $J_{f_\theta}(z^*)$ would prevent simpler iterative solvers from diverging or oscillating. [9] introduce a Jacobian term to the training objective to regularize the model training. The authors use the Hutchinson estimator [23] to compute and regularize the Frobenius norm of the Jacobian.

$$\mathcal{L}_{jac} = \lambda_{jac} \frac{\|\epsilon^T J_{f_\theta}(z^*)\|_2^2}{d}; \quad \epsilon \sim \mathcal{N}(0, I_d)$$

While well-motivated, the disadvantage of this method is that the Hutchinson trace estimator requires automatic differentiation in the loss function, thus requiring higher order differentiation in the training process and greatly increasing the training costs. However, in return for the increased cost, it was demonstrated that increased robustness followed, along with faster forward passes in the trained results. Our methods are orthogonal to the Jacobian stabilization process. In Section IV, we provide empirical evidence on composing our models with Jacobian Stabilization to achieve even more robust results.

III. Methods

A. Continuous Deep Equilibrium Networks

Deep Equilibrium Models have traditionally been formulated as steady-state problems for a discrete dynamical system. However, discrete dynamical systems come with a variety of shortcomings. Consider the following linear discrete dynamical system (See Figure 2):

$$u_{n+1} = \alpha \cdot u_n \qquad \texttt{where } \|\alpha\| < 1 \texttt{ and } u_0 = 1$$

This system converges to a steady state of $u_\infty = 0$. However, in many cases, this convergence can be relatively slow. If $\alpha = 0.9$, then after 10 steps, the value is $u_{10} = 0.35$ because a small amount only reduces each successive step. Thus convergence could only be accelerated by taking many steps together. Even further, if $\alpha = -0.9$, the value ping-pongs over the steady state $u_1 = -0.9$, meaning that if we could take some fractional step $u_{\delta t}$ then it would be possible to approach the steady state much faster. [24, 25] describe several other shortcomings of using discrete steady-state dynamics over continuous steady-state dynamics. These issues combined motivate changing from a discrete description of the system (the fixed point or Broyden's method approach) to a continuous description of the system that allows adaptivity to change the stepping behavior and accelerate convergence.

To this end, we propose an alternate formulation for DEQs by modeling a continuous dynamical system (Continuous DEQ) where the forward pass is represented by an ODE which is solved from $t_0 = 0$ to $t_1 = \infty$:

$$\frac{dz}{dt} = f_\theta(z, x) - z$$

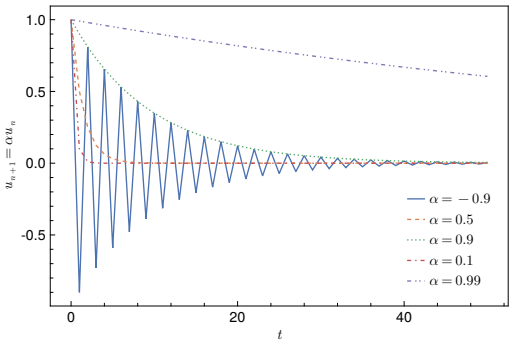

Fig. 2: **Slow Convergence of Simple Linear Discrete Dynamical Systems**

where f_θ is an explicit neural network. Continuous DEQs leverage fast adaptive ODE solvers, which terminate automatically once the solution is close to a steady state, i.e., $\frac{dz^*}{dt} = 0$, which then satisfies $f_\theta(z^*, x) = z^*$ and is thus the solution to the same implicit system as before.

The Continuous DEQ can be considered an infinite-time neural ODE in this form. However, almost paradoxically, the infinite time version is cheaper to train than the finite time version as its solution is the solution to the nonlinear system, meaning the same implicit differentiation formula of the original DEQ holds for the derivative. This means that no backpropagation through the steps is required for the Continuous DEQ, and only a linear system must be solved. In Section IV, we empirically demonstrate that Continuous DEQs outperform Neural ODEs in terms of training time while achieving similar accuracies.

B. Skip Deep Equilibrium Networks

[2, 8] set the initial condition $u_0 = 0$ while solving a DEQ. Assuming the existence of a steady state, the solvers will converge given enough iterations. However, each iteration is expensive, and a poor guess of the initial condition makes the convergence slower. To counteract these issues, we introduce an alternate architecture for DEQ (Skip DEQ), where we use an explicit model g_ϕ to predict the initial condition for the steady-state problem $u_0 = g_\phi(x)$[2]. We jointly optimize for $\{\theta, \phi\}$ by adding an auxiliary loss function:

$$\mathcal{L}_{\texttt{skip}} = \lambda_{\texttt{skip}} \|f_\theta(z^*, x) - g_\phi(x)\|$$

Intuitively, our explicit model g_ϕ better predicts a value closer to the steady-state (over the training iterations), and hence we need to perform fewer iterations during the forward pass. Given that its prediction is relatively free compared to the cost of the DEQ, this technique could decrease the cost of the DEQ by reducing the total number of iterations required. However, this prediction-correction approach still uses the result of the DEQ for its final predictions and thus should achieve robustness properties equal.

[2]We note that the concurrent work [26] introduced a similar formulation as a part of HyperDEQ

Model	Jacobian Reg.	# of Params	Test Accuracy (%)	Testing NFE	Training Time (min)	Prediction Time (s / batch)
Vanilla DEQ	✗	138K	97.926 ± 0.107	18.345 ± 0.732	5.197 ± 1.106	0.038 ± 0.009
	✓		98.123 ± 0.025	5.034 ± 0.059	7.321 ± 0.454	0.011 ± 0.005
Skip DEQ	✗	151K	97.759 ± 0.080	4.001 ± 0.001	1.711 ± 0.202	0.010 ± 0.001
	✓		97.749 ± 0.141	4.001 ± 0.000	6.019 ± 0.234	0.012 ± 0.001
Skip Reg. DEQ	✗	138K	97.973 ± 0.134	4.001 ± 0.000	1.295 ± 0.222	0.010 ± 0.001
	✓		98.016 ± 0.049	4.001 ± 0.000	5.128 ± 0.241	0.012 ± 0.000

TABLE I: **MNIST Classification with Fully Connected Layers**: Skip Reg. Continuous DEQ without Jacobian Regularization takes $4\times$ *less training time* and *speeds up prediction time by* $4\times$ compared to Continuous DEQ. Continuous DEQ with Jacobian Regularization has a similar prediction time but takes $6\times$ *more training time* than Skip Reg. Continuous DEQ. Using Skip variants *speeds up training by* $1.42 \times -4\times$.

1) Skip Regularized DEQ: Regularization Scheme without Extra Parameters: One of the primary benefits of DEQs is the low memory footprint of these models (See Section II). Introducing an explicit model g_ϕ increases the memory requirements for training. To alleviate this problem, we propose a regularization term to minimize the L1 distance between the first prediction of f_θ and the steady-state solution:

$$\mathcal{L}_{\texttt{skip}} = \lambda_{\texttt{skip}} \| f_\theta(z^*, x) - f_\theta(0, x) \|$$

This technique follows the same principle as the Skip DEQ where the DEQ's internal neural network is now treated as the prediction model. We hypothesize that this introduces an inductive bias in the model to learn simpler training dynamics.

IV. Experiments

In this section, we consider the effectiveness of our proposed methods – Continuous DEQs and Skip DEQs – on the training and prediction timings. We consider the following baselines:

1) Discrete DEQs with L-Broyden Solver.
2) Jacobian Regularization of DEQs.[3]
3) Multi-Scale Neural ODEs with Input Injection: A modified Continuous Multiscale DEQ without the steady state convergence constaint.

Our primary metrics are classification accuracy, the number of function evaluations (NFEs), total training time, time for the backward pass, and prediction time per batch. We showcase the performance of our methods on – MNIST [27], CIFAR-10 [28], SVHN [29], & ImageNet [30]. We use perform our experiments in Julia [31] using Lux.jl [32] and DifferentialEquations.jl [33, 34, 35]. We provide the exact experimental setup in Appendix A, B, & C.

A. MNIST Image Classification

We summarize our results in Table I. Without Jacobian Stabilization, Skip Reg. Continuous DEQ has the highest testing accuracy of 97.973% and has the *lowest training and prediction timings overall*. Using Jacobian Regularization, DEQ outperforms Skip DEQ models by $< 0.4\%$, however, jacobian regularization increases training time by $1.4 - 4\times$. Skip DEQ models can obtain the lowest prediction time per batch of $\sim 0.01s$.

B. CIFAR10 Image Classification

We summarize our results for the smaller 200K parameter model in Table II and Figure 3. Continuous DEQs are faster than Neural ODEs during training by a factor of $2 \times -2.36\times$, with a speedup of $4.2 \times -8.67\times$ in the backward pass.

We summarize our 11M parameter model results in Table III and Figure 4. Continuous DEQs are faster than Neural ODEs during training by a factor of $4.1 \times -7.98\times$, with a speedup of $6.18 \times -36.552\times$ in the backward pass.

C. ImageNet Image Classification

We summarize our results in Table IV and Figure 5. Skip (Reg.) variants accelerate the training of Continuous DEQ by $1.57 \times -1.96\times$, with a reduction of $2.2 \times -4.2\times$ in the backward pass timings.

V. Related Works

A. Implicit Models

Implicit Models have obtained competitive results in image processing [8], generative modeling [4], time-series prediction [14], etc, at a fraction of memory requirements for explicit models. Additionally, [36] show that for a certain class of DEQs convergence to global optima is guaranteed at a linear rate. However, the slow training and prediction timings [5, 6, 7, 9, 13, 37] often overshadow these benefits.

B. Accelerating Neural ODEs

[6, 7] used higher-order regularization terms to constrain the space of learnable dynamics for Neural ODEs. Despite speeding up predictions, these models often increase the training time by 7x [37]. Alternatively, [13] randomized the endpoint of Neural ODEs to incentivize simpler dynamics. [37] used internal solver heuristics – local error and stiffness estimates – to control the learned dynamics in a way that decreased both prediction and training time. [38] rewrite Neural ODEs as heavy ball ODEs to accelerate both forward and backward passes. [39] replace ODE solvers in the forward with a Taylor-Lagrange expansion and report significantly better training and prediction times.

Regularized Neural ODEs can not be directly extended to discrete DEQs [2, 8]. Our continuous formulation introduces the

[3]We note that due to limitations of our Automatic Differentiation system, we cannot perform Jacobian Regularization for Convolutional Models. However, our preliminary analysis suggests that the Skip DEQ and Continuous DEQ approaches are fully composable with Jacobian Regularization and provide better performance compared to using only Jacobian Regularization (See Table I).

Model	Continuous	# of Params	Test Accuracy (%)	Training Time (s / batch)	Backward Pass (s / batch)	Prediction Time (s / batch)
Vanilla DEQ	✗	163546	81.233 ± 0.097	0.651 ± 0.009	0.075 ± 0.001	0.282 ± 0.005
	✓		80.807 ± 0.631	0.753 ± 0.017	0.261 ± 0.010	0.136 ± 0.010
Skip DEQ	✗	200122	82.013 ± 0.306	0.717 ± 0.022	0.115 ± 0.004	0.274 ± 0.005
	✓		80.807 ± 0.230	0.806 ± 0.010	0.293 ± 0.004	0.154 ± 0.002
Skip Reg. DEQ	✗	163546	81.170 ± 0.356	0.709 ± 0.005	0.114 ± 0.002	0.283 ± 0.007
	✓		82.513 ± 0.177	0.679 ± 0.015	0.143 ± 0.017	0.154 ± 0.003
Neural ODE	✓	163546	83.543 ± 0.393	1.608 ± 0.026	1.240 ± 0.021	0.207 ± 0.006

TABLE II: **CIFAR10 Classification with Small Neural Network**: Skip Reg. Continuous DEQ achieves the *highest test accuracy among DEQs*. Continuous DEQs are faster than Neural ODEs during training by a factor of $2 \times -2.36\times$, with a speedup of $4.2 \times -8.67\times$ in the backward pass. We also observe a prediction speed-up for Continuous DEQs of $1.77 \times -2.07\times$ against Discrete DEQs and $1.34 \times -1.52\times$ against Neural ODE.

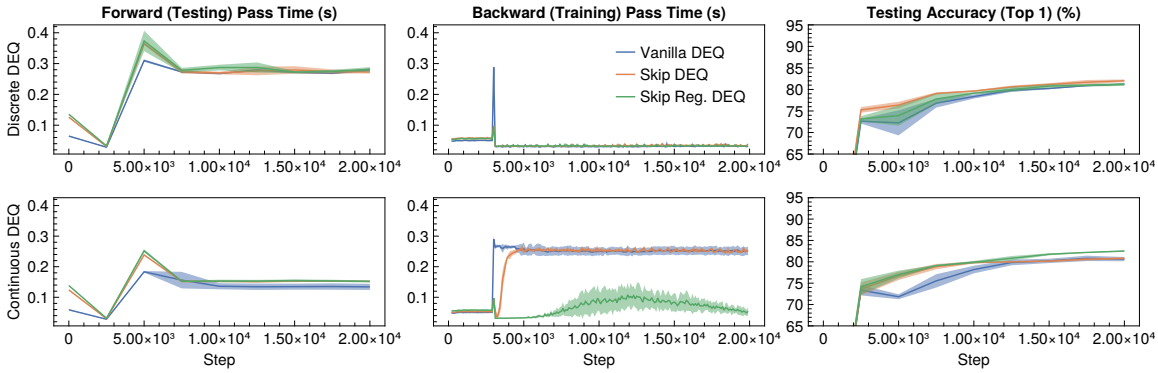

Fig. 3: **CIFAR10 Classification with Small Neural Network**

Model	Continuous	# of Params	Test Accuracy (%)	Training Time (s / batch)	Backward Pass (s / batch)	Prediction Time (s / batch)
Vanilla DEQ	✗	10.63M	88.913 ± 0.287	0.625 ± 0.165	0.111 ± 0.021	0.414 ± 0.222
	✓		89.367 ± 0.832	1.284 ± 0.011	0.739 ± 0.003	0.606 ± 0.010
Skip DEQ	✗	11.19M	88.783 ± 0.178	0.588 ± 0.042	0.112 ± 0.006	0.314 ± 0.017
	✓		89.600 ± 0.947	0.697 ± 0.012	0.150 ± 0.013	0.625 ± 0.004
Skip Reg. DEQ	✗	10.63M	88.773 ± 0.115	0.613 ± 0.048	0.109 ± 0.008	0.268 ± 0.031
	✓		90.107 ± 0.837	0.660 ± 0.019	0.125 ± 0.003	0.634 ± 0.019
Neural ODE	✓	10.63M	89.047 ± 0.116	5.267 ± 0.078	4.569 ± 0.077	0.573 ± 0.010

TABLE III: **CIFAR10 Classification with Large Neural Network**: Skip Reg. Continuous DEQ achieves the *highest test accuracy*. Continuous DEQs are faster than Neural ODEs during training by a factor of $4.1 \times -7.98\times$, with a speedup of $6.18 \times -36.552\times$ in the backward pass. However, we observe a prediction slowdown for Continuous DEQs of $1.4 \times -2.36\times$ against Discrete DEQs and $0.90 \times -0.95\times$ against Neural ODE.

potential to extend [38, 39] to DEQs. However, these methods benefit from the structure in the backward pass, which does not apply to DEQs. Additionally, relying on discrete sensitivity analysis [37] nullifies the benefit of a cost-effective backward pass.

C. Accelerating DEQs

[9] uses second-order derivatives to regularize the Jacobian, stabilizing the training and prediction timings of DEQs. [40] proposes a Jacobian-Free Backpropagation Model, which accelerates solving the Linear Equation in the backward pass. Our work complements these models and can be freely composed with them. We have shown that a poor initial condition harms convergence, and a better estimate for the same leads to faster training and prediction. We hypothesize that combining these methods would lead to more stable and faster convergence, demonstrating this possibility with the Jacobian regularization Skip DEQ.

VI. Discussion

We have empirically shown the effectiveness of Continuous DEQs as a faster alternative for Neural ODEs. Consistent

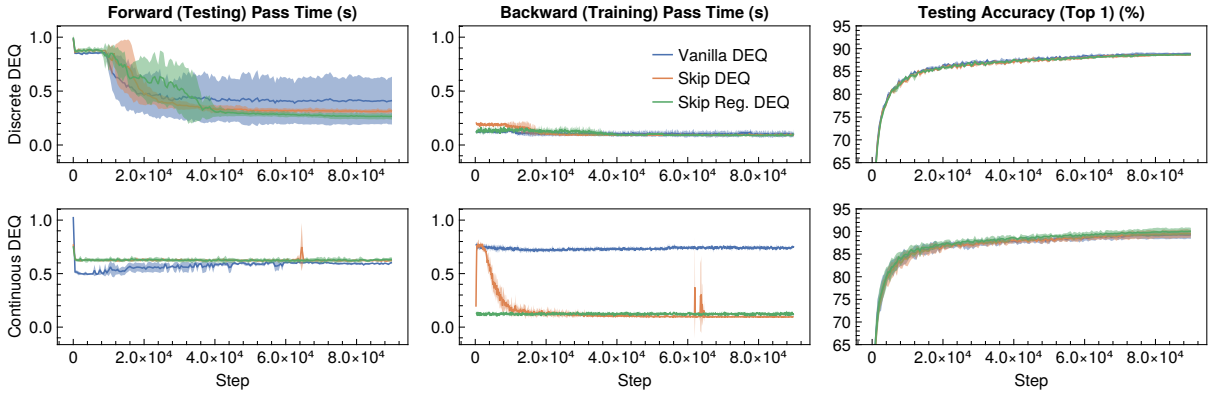

Fig. 4: **CIFAR10 Classification with Large Neural Network**

Model	Continuous	# of Params	Test Accuracy (Top 5) (%)	Training Time (s / batch)	Backward Pass (s / batch)	Prediction Time (s / batch)
Vanilla DEQ	✗	17.91M	81.809 ± 0.115	2.057 ± 0.138	0.195 ± 0.007	1.963 ± 0.189
	✓		81.329 ± 0.516	3.131 ± 0.027	1.873 ± 0.015	1.506 ± 0.027
Skip DEQ	✗	18.47M	81.717 ± 0.452	1.956 ± 0.012	0.194 ± 0.001	1.843 ± 0.025
	✓		81.334 ± 0.322	2.016 ± 0.129	0.845 ± 0.127	1.575 ± 0.053
Skip Reg. DEQ	✗	17.91M	81.611 ± 0.369	1.996 ± 0.035	0.539 ± 0.023	1.752 ± 0.093
	✓		81.813 ± 0.350	1.607 ± 0.044	0.444 ± 0.026	1.560 ± 0.021

TABLE IV: **ImageNet Classification**: All the variants attain comparable evaluation accuracies. Skip (Reg.) accelerates the training of Continuous DEQ by $1.57\times - 1.96\times$, with a reduction of $2.2\times - 4.2\times$ in the backward pass timings. However, we observe a marginal increase of 4% in prediction timings for Skip (Reg.) Continuous DEQ compared against Continuous DEQ. For Discrete DEQs, Skip (Reg.) variants reduce the prediction timings by $6.5\% - 12\%$.

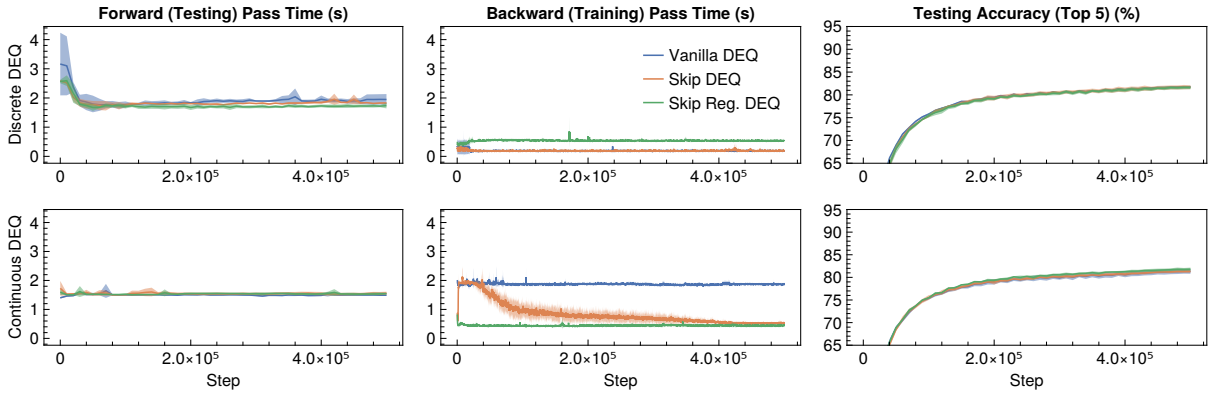

Fig. 5: **ImageNet Classification**

with the ablation studies in [26], we see that Skip DEQ in itself doesn't significantly improve the prediction or training timings for Discrete DEQs. Skip Reg. DEQ does, however, speeds up the inference for larger Discrete DEQs. However, combining Skip DEQ and Skip Reg. DEQ with Continuous DEQs, enable a speedup in backward pass by over $2.8 - 5.9\times$. We hypothesize that this improvement is due to reduction in the condition number, which results in faster convergence of GMRES in the backward pass, however, acertaining this would require furthur investigation. We have demonstrated that our improvements to DEQs and Neural ODEs enable the drop-in replacement of Skip Continuous DEQs in any classical deep learning problem where continuous implicit models were previously employed.

A. *Limitations*

We observe the following limitations for our proposed methods:

- Reformulating a Neural ODE as a Continuous DEQ is valid, when the actual dynamics of the system doesn't matter. This holds true for all applications of Neural ODEs to classical Deep Learning problems.
- Continuous DEQs are slower than their Discrete counterparts for larger models (without any significant improvement to accuracy), hence the authors recommend their usage only for cases where a continuous model is truly needed.

VII. Acknowledgement

The authors acknowledge the MIT SuperCloud and Lincoln Laboratory Supercomputing Center for providing HPC resources that have contributed to the research results reported within this paper. This material is based upon work supported by the National Science Foundation under grant no. OAC-1835443, grant no. SII-2029670, grant no. ECCS-2029670, grant no. OAC-2103804, and grant no. PHY-2021825. We also gratefully acknowledge the U.S. Agency for International Development through Penn State for grant no. S002283-USAID. The information, data, or work presented herein was funded in part by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award Number DE-AR0001211 and DE-AR0001222. We also gratefully acknowledge the U.S. Agency for International Development through Penn State for grant no. S002283-USAID. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. This material was supported by The Research Council of Norway and Equinor ASA through Research Council project "308817 - Digital wells for optimal production and drainage". Research was sponsored by the United States Air Force Research Laboratory and the United States Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[1] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," *arXiv preprint arXiv:1806.07366*, 2018.

[2] S. Bai, J. Z. Kolter, and V. Koltun, "Deep Equilibrium Models," *arXiv:1909.01377 [cs, stat]*, Oct. 2019. arXiv: 1909.01377.

[3] L. E. Ghaoui, F. Gu, B. Travacca, A. Askari, and A. Y. Tsai, "Implicit Deep Learning," *arXiv:1908.06315 [cs, math, stat]*, Aug. 2020. arXiv: 1908.06315.

[4] W. Grathwohl, R. T. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, "Ffjord: Free-form continuous dynamics for scalable reversible generative models," *arXiv preprint arXiv:1810.01367*, 2018.

[5] E. Dupont, A. Doucet, and Y. W. Teh, "Augmented neural odes," *arXiv preprint arXiv:1904.01681*, 2019.

[6] J. Kelly, J. Bettencourt, M. J. Johnson, and D. Duvenaud, "Learning differential equations that are easy to solve," *arXiv preprint arXiv:2007.04504*, 2020.

[7] C. Finlay, J.-H. Jacobsen, L. Nurbekyan, and A. M. Oberman, "How to train your neural ode," *arXiv preprint arXiv:2002.02798*, 2020.

[8] S. Bai, V. Koltun, and J. Z. Kolter, "Multiscale Deep Equilibrium Models," *arXiv:2006.08656 [cs, stat]*, Nov. 2020. arXiv: 2006.08656.

[9] S. Bai, V. Koltun, and J. Z. Kolter, "Stabilizing equilibrium models by jacobian regularization," *arXiv preprint arXiv:2106.14342*, 2021.

[10] X. Liu, S. Si, Q. Cao, S. Kumar, and C.-J. Hsieh, "Neural sde: Stabilizing neural ode networks with stochastic noise," *arXiv preprint arXiv:1906.02355*, 2019.

[11] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *International Conference on Machine Learning*, pp. 136–145, PMLR, 2017.

[12] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter, "Differentiable convex optimization layers," *arXiv preprint arXiv:1910.12430*, 2019.

[13] A. Ghosh, H. S. Behl, E. Dupont, P. H. Torr, and V. Namboodiri, "Steer: Simple temporal regularization for neural odes," *arXiv preprint arXiv:2006.10711*, 2020.

[14] Y. Rubanova, R. T. Chen, and D. Duvenaud, "Latent odes for irregularly-sampled time series," *arXiv preprint arXiv:1907.03907*, 2019.

[15] C. G. Broyden, "A class of methods for solving nonlinear simultaneous equations," *Mathematics of computation*, vol. 19, no. 92, pp. 577–593, 1965.

[16] S. G. Johnson, "Notes on adjoint methods for 18.335," *Introduction to Numerical Methods*, 2006.

[17] Y. Saad and M. H. Schultz, "Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM Journal on scientific and statistical computing*, vol. 7, no. 3, pp. 856–869, 1986.

[18] P. J. Burt and E. H. Adelson, "The laplacian pyramid as a compact image code," in *Readings in computer vision*, pp. 671–679, Elsevier, 1987.

[19] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1915–1929, 2012.

[20] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[21] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, "Attention to scale: Scale-aware semantic image segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3640–3649, 2016.

[22] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[23] M. F. Hutchinson, "A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines," *Communications in Statistics-Simulation and Computation*, vol. 18, no. 3, pp. 1059–1076, 1989.

[24] R. Rico-Martinez, K. Krischer, I. Kevrekidis, M. Kube, and J. Hudson, "Discrete-vs. continuous-time nonlinear signal processing of cu electrodissolution data," *Chemical Engineering Communications*, vol. 118, no. 1, pp. 25–48, 1992.

[25] A. Bulsari, *Neural Networks for Chemical Engineers*. Computer-aided chemical engineering, Elsevier, 1995.

[26] S. Bai, V. Koltun, and J. Z. Kolter, "Neural deep equilibrium solvers," in *International Conference on Learning Representations*, 2021.

[27] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[28] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[29] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[31] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Review*, vol. 59, no. 1, pp. 65–98, 2017.

[32] A. Pal, "Lux: Explicit parameterization of deep neural networks in julia." https://github.com/avik-pal/Lux.jl/, 2022.

[33] C. Rackauckas and Q. Nie, "Differentialequations.jl – a performant and feature-rich ecosystem for solving differential equations in julia," *The Journal of Open Research Software*, vol. 5, no. 1, 2017. Exported from https://app.dimensions.ai on 2019/05/05.

[34] C. Rackauckas, Y. Ma, V. Dixit, X. Guo, M. Innes, J. Revels, J. Nyberg, and V. Ivaturi, "A comparison of automatic differentiation and continuous sensitivity analysis for derivatives of differential equation solutions," *arXiv preprint arXiv:1812.01892*, 2018.

[35] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, and A. Edelman, "Universal differential equations for scientific machine learning," *arXiv preprint arXiv:2001.04385*, 2020.

[36] K. Kawaguchi, "On the theory of implicit deep learning: Global convergence with implicit layers," *arXiv preprint arXiv:2102.07346*, 2021.

[37] A. Pal, Y. Ma, V. Shah, and C. V. Rackauckas, "Opening the blackbox: Accelerating neural differential equations by regularizing internal solver heuristics," in *Proceedings of the 38th International Conference on Machine Learning*, vol. 139 of *Proceedings of Machine Learning Research*, pp. 8325–8335, PMLR, 18–24 Jul 2021.

[38] H. Xia, V. Suliafu, H. Ji, T. Nguyen, A. Bertozzi, S. Osher, and B. Wang, "Heavy ball neural ordinary differential equations," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[39] F. Djeumou, C. Neary, E. Goubault, S. Putot, and U. Topcu, "Taylor-lagrange neural ordinary differential equations: Toward fast training and evaluation of neural odes," 2022.

[40] S. W. Fung, H. Heaton, Q. Li, D. McKenzie, S. Osher, and W. Yin, "Jfb: Jacobian-free backpropagation for implicit networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[41] C. Tsitouras, "Runge–kutta pairs of order 5 (4) satisfying only the first column simplifying assumption," *Computers & Mathematics with Applications*, vol. 62, no. 2, p. 770–775, 2011.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[43] G. Wanner and E. Hairer, *Solving ordinary differential equations II*, vol. 375. Springer Berlin Heidelberg New York, 1996.

[44] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

APPENDIX

A. MNIST Experimental Details

Training Details: Following [6], our Fully Connected Model consists of 3 layers – a downsampling layer $\mathbb{R}^{784} \mapsto \mathbb{R}^{128}$, continuous DEQ layer $f_\theta : \mathbb{R}^{128} \mapsto \mathbb{R}^{128}$, and a linear classifier $\mathbb{R}^{128} \mapsto \mathbb{R}^{10}$.

For regularization, we use $\lambda_{\texttt{skip}} = 0.01$ and train the models for 25 epochs with a batch size of 32. We use Tsit5 [41] with a relative tolerance for convergence of 0.005. We use Adam [42] with a constant learning rate of 0.001 for optimization.

Baselines: We use continuous DEQ and continuous DEQ with Jacobian Stabilization as our baselines. We additionally compose Skip DEQs with Jacobian Stabilization in our benchmarks. For all experiments, we keep $\lambda_{\texttt{jac}} = 1.0$.

B. CIFAR10 Experimental Details

For all the baselines in this section, Vanilla DEQ is trained with the same training hyperparameters as the corresponding Skip DEQs (taken from [8]). Multiscale Neural ODE with Input Injection is trained with the same hyperparameters as the corresponding Continuous DEQs.

1) Architecture with 200K parameters: **Training Details:** Our Multiscale DEQ architecture is the same as MDEQ-small architecture used in [8]. For the explicit network in Skip DEQ, we use the residual block and downsampling blocks from [8] which account for the additional 58K trainable parameters.

We use a fixed regularization weight of $\lambda_{\texttt{skip}} = 0.01$ and the models are trained for 20000 steps. We use a batch size of 128. For continuous models, we use VCAB3 [43] with a relative tolerance for convergence of 0.05. We use AdamW [44] optimizer with a cosine scheduling on the learning rate – starting from 10^{-3} and terminating at 10^{-6} – and a weight decay of 2.5×10^{-6}.

2) Architecture with 11M parameters: **Training Details:** Our Multiscale DEQ architecture is the same as MDEQ-large architecture used in [8]. For the explicit network in Skip DEQ, we use the residual block and downsampling blocks from [8] which account for the additional 58K trainable parameters.

We use a fixed regularization weight of $\lambda_{\texttt{skip}} = 0.01$ and the models are trained for 90000 steps. We use a batch size of 128. For continuous models, we use VCAB3 [43] with a relative tolerance for convergence of 0.05. We use Adam [42] optimizer with a cosine scheduling on the learning rate – starting from 10^{-3} and terminating at 10^{-6}.

C. ImageNet Experimental Details

Training Details: Our Multiscale DEQ architecture is the same as MDEQ-small architecture used in [8]. For the explicit network in Skip DEQ, we use the residual block and downsampling blocks from [8] which account for the additional 58K trainable parameters.

We use a fixed regularization weight of $\lambda_{\texttt{skip}} = 0.01$, and the models are trained for 500000 steps. We use a batch size of 64. For continuous models, we use VCAB3 [43] with a relative tolerance for convergence of 0.05. We use SGD with a momentum of 0.9 and weight decay of 10^{-6}. We use a step LR scheduling reducing the learning rate from 0.05 by a multiplicative factor of 0.1 at steps 100000, 150000, and 250000.

Baselines: Vanilla DEQ is trained with the same training hyperparameters as the corresponding Skip DEQs (taken from [8])[4].

[4]When training MultiScale Neural ODE with the same configuration as Continuous DEQ, we observed a $8\times$ slower backward pass which made the training of the baseline infeasible.