Optimization Induced Equilibrium Networks: An Explicit Optimization Perspective for Understanding Equilibrium Models

Xingyu Xie, *Student Member, IEEE*, Qiuhao Wang, Zenan Ling, Xia Li, Guangcan Liu, *Senior Member, IEEE*, and Zhouchen Lin, *Fellow, IEEE*

Abstract—To reveal the mystery behind deep neural networks (DNNs), optimization may offer a good perspective. There are already some clues showing the strong connection between DNNs and optimization problems, e.g., under a mild condition, DNN's activation function is indeed a proximal operator. In this paper, we are committed to providing a unified optimization induced interpretability for a special class of networks—equilibrium models, i.e., neural networks defined by fixed point equations, which have become increasingly attractive recently. To this end, we first decompose DNNs into a new class of unit layer that is the proximal operator of an implicit convex function while keeping its output unchanged. Then, the equilibrium model of the unit layer can be derived, we name it Optimization Induced Equilibrium Networks (OptEq). The equilibrium point of OptEq can be theoretically connected to the solution of a convex optimization problem with explicit objectives. Based on this, we can flexibly introduce prior properties to the equilibrium points: 1) modifying the underlying convex problems explicitly so as to change the architectures of OptEq; and 2) merging the information into the fixed point iteration, which guarantees to choose the desired equilibrium point when the fixed point set is non-singleton. We show that OptEq outperforms previous implicit models even with fewer parameters.

Index Terms—Equilibrium models, DEQ, optimization induced models, interpretability

1 Introduction

For a long time, DNNs have been regarded as powerful "black boxes" but lacking interpretability. Roughly, all we know is that different DNNs contain some specific inductive biases, such as convolutional structures aiming to extract local features, while Transformers can model global semantics more easily. However, a more detailed way to quantify this different bias remains missing. The implicit models have recently gained significant attention due to their comparable performance and much less memory-consuming. Instead of specifying the explicit forward procedure, the implicit model specifies some conditions held at its output. For Neural ODE [1], the neural network is replaced by an ordinary differential equation (ODE) flow, which can be seen as a continuous version of ResNets [2]. Another instance of such models is Deep Equilibrium Model (DEQ) [3], [4], which observed that, in the weight-tying case (share the weights in each layer), neural network's forward propagation is equivalent to the procedure of power iteration, and its output approaches the fixed point of a single layer. Thus the equilibrium model's output is created by finding a solution to the fixed point equation. Although having a simpler formulation, equilibrium model still lacks interpretability in some sense. Because the underlying meaning of the fixed point equation is unclear, therefore we still cannot analyze its properties and understand the intrinsic mechanisms.

It is interesting to provide interpretability for neural networks from the optimization perspective. In fact, there are already some clues showing that DNNs' components are strongly related to some underlying optimization problems. [5] shows that DNN's activation function is indeed a proximal operator if it is non-decreasing. [6] investigates the potential energy function of the self-attention operators. [7] reformulates the single layer of feed-forward NNs as an argument minimum operator, therefore performing forward propagation becomes solving several constrained convex problems. On the other hand, there is limited work to show the optimization connection for equilibrium models. Previous work [8], [9] proved that equilibrium model with

- *Xingyu Xie, Qiuhao Wang, and Zenan Ling are with the Key Laboratory of Machine Perception (MOE), School of Artificial Intelligence, Peking University, Beijing 100871, China. E-mail: {xyxie, wqh19, lingzenan}@pku.edu.cn.*
- *Xia Li is with the Swiss Federal Institute of Technology, 8092 Zürich, Switzerland. E-mail: xiaxiali@ethz.ch.*
- *Guangcan Liu is with the School of Automation, Southeast University, Nanjing 210018, China. E-mail: gcliu1982@gmail.com.*
- *Zhouchen Lin is with the Key Laboratory of Machine Perception (MOE), School of Artificial Intelligence, Peking University, Beijing 100871, China, and with the Institute for Artificial Intelligence, Peking University, Beijing 100871, China, and also with Peng Cheng Laboratory, Shenzhen 518066, China. E-mail: zlin@pku.edu.cn.*

Manuscript received 14 Dec. 2021; revised 27 Apr. 2022; accepted 2 June 2022. Date of publication 10 June 2022; date of current version 3 Feb. 2023.
This work was supported in part by the New Generation AI Major Project of Ministry of Science and Technology of China under Grant 2018AAA0102501, in part by the National Natural Science Foundation of China (NSFC) under Grant U21B2027. The work of Zhouchen Lin was supported in part by the major key project of PCL under Grant PCL2021A12, in part by NSF China under Grant 61731018, in part by NSFC the Tianyuan Fund for Mathematics under Grants 12026606, and 2020BD006, and supported in part by PKU-Baidu Fund.
(Corresponding author: Zhouchen Lin.)
Recommended for acceptance by M. Long.
Digital Object Identifier no. 10.1109/TPAMI.2022.3181425

the form $\mathbf{z}^* = \sigma(\mathbf{W}\mathbf{z}^* + \mathbf{U}\mathbf{x} + \mathbf{b})$ is equivalent to an operator splitting problem, but the operator splitting formulation does not have good physical interpretation and it can only be associated with an optimization problem under restricted conditions (e.g., ReLU activation and p.s.d. weight [9]).

Suppose that the equilibrium model can connect with an underlying optimization problem and the explicit objective exists. We may further understand the mechanism of the equilibrium model and the reasons for its empirical success. For example, we can introduce the customized property into the equilibrium model architectures by adding some regularization terms to the underlying optimization objective, which may help to inspire more equilibrium model architectures. Moreover, when finding the model's output is equivalent to minimizing a convex objective, we can utilize any optimization algorithm, especially the accelerated ones, to obtain the equilibrium point rather than the fixed-point iteration and the root find methods, which are heavily used in the previous equilibrium models.

To understand the equilibrium models from the optimization perspective, this paper investigates a new equilibrium model— Optimization Induced Equilibrium Networks (OptEq), which can be well interpreted from the optimization perspective: solving OptEq's fixed point equation is equivalent to getting the minimizer of an underlying optimization problem. We first show that OptEq's structure can be naturally extracted from any feed-forward NN in Lemma 1. Specially, we reformulate the general feed-forward NN and decompose it into the composition of a new class of unit layers while keeping the output unchanged. We then reveal that OptEq's layer is the proximal operator of an underlying implicit convex function in Theorem 1. Our main contribution comes from two aspects: (1) we discover the intrinsic optimization-induced structure hidden in general feed-forward NNs; (2) the proposed new equilibrium model can be well understood from the optimization perspective. *In principle, we can customize the equilibrium model* by modifying the underlying optimization problem. For example, OptEq naturally produces the commonly used skip connection architecture by replacing the convex objective with its Moreau envelope, which keeps the equilibrium point unchanged. To strengthen the representation ability, we propose a deep version of OptEq, which includes the general feed-forward DNN as a special case, whose underlying convex objective is the sum of single ones of its contained layers. Finally, we utilize OptEq's optimization induced property to improve the equilibrium model further. We offer two methods to introduce any customized prior information to the equilibrium of OptEq. The first way is to modify the underlying optimization problem directly, which leads to the changes of OptEq's architecture. Another way is to use a modified SAM [10] iteration for selecting the fixed point with the minimal regularization values when the fixed point set is non-singleton. In summary, our contributions include:

- By decomposing the general DNN, we discover the intrinsic optimization-induced layer hidden in general feed-forward NNs, which is the proximal operator of a convex function. Then we extract the layer to make it an equilibrium model called OptEq, and further propose its deep version. Without further reparameterization, the equilibrium point of OptEq is a solution to an underlying convex problem. So OptEq's property can be well-studied by investigating the underlying optimization problem.
- We propose two methods to introduce customized properties to the model's equilibrium points. One is inspired by the underlying optimization, and the other is induced by a modified SAM [11] iteration. This is the first time that we can customize equilibrium models in a principled way.
- We conduct experiments on CIFAR-10 and ImageNet for image classification and Cityscapes for semantic segmentation. Deep OptEq significantly outperforms baseline equilibrium models. Moreover, we also provide several feature regularizations that can significantly improve the generalization.

2 Related Work

2.1 Deep Equilibrium Models

Bai *et al.* [3] pioneered the DEQ, an entirely implicit neural network, by replacing the DNN's forward propagation with a fixed point equation, which can be regarded as a weight-tied DNN with infinite layers. DEQ solves its fixed point equation by the quasi-Newton method and back-propagates itself with implicit differentiation. Many works have investigated DEQs from different perspectives. For example, previous work [8], [9] was devoted to ensuring the stability of DEQ, i.e., the existence and uniqueness of fixed point, and proved that DEQ is equivalent to an operator splitting problem under some reparameterization. There is also some work considering the training strategies [12], [13], and training dynamics [14] for DEQs.

The most relevant work is [9], which associates DEQ with an optimization problem under very restricted conditions (ReLU activation and p.s.d. weight matrix). By contrast, in this work, we try to understand equilibrium models from the optimization perspective, and hence propose our OptEq, which naturally has an underlying optimization objective function under quite mild conditions. Based on this, we suggest principled ways to incorporate prior information into OptEq.

2.2 Bilevel Optimization

Bilevel optimization is a popular research area in optimization and has a wide range of applications in machine learning, like Hyperparameter Optimization [15], Meta-Learning [16], and Neural Architecture Search [17]. Equilibrium models training has many similarities with solving a bilevel optimization problem [18]. For the proposed OptEq, whose output is the minimizer of a convex function exactly, model training is equivalent to solving a bilevel optimization problem. In specific, for training Opteq, we have

$$\begin{array}{ll} \min_{\boldsymbol{\theta}} \operatorname{loss}(\mathbf{z}^*(\mathbf{x},\boldsymbol{\theta}),\mathbf{y}), \\ s.t.\ \mathbf{z}^* \in \operatorname{Fix}(\mathcal{T}(\mathbf{z},\boldsymbol{\theta})), \end{array} \quad \Leftrightarrow \quad \begin{array}{ll} \min_{\boldsymbol{\theta}} \operatorname{loss}(\mathbf{z}^*(\mathbf{x},\boldsymbol{\theta}),\mathbf{y}), \\ s.t.\ \mathbf{z}^* \in \operatorname{argmin}_{\mathbf{z}} \Phi(\mathbf{z},\boldsymbol{\theta}). \end{array}$$

Note that, for genetal DEQ, this transformation is ill-defined, since the lower-level optimization objective function $\Phi(\cdot)$

does not exist in most cases. Inspired by the well-known SAM algorithm [11] in the bilevel optimization community, we propose the unrolling SAM method to perform OptEq training in Section 5.2.

Note that in Section 5.3, we discuss the difference between the implicit differentiation method and the unrolling algorithm method, which are two important methods for solving bilevel optimization problems [16], [19]. A more detailed discussion of the advantages and disadvantages of these two types of methods can be found in the survey [20]. In practice, the appropriate algorithm should be chosen based on the trade-off between memory and speed.

2.3 Optimization-Inspired Neural Networks

In general, Optimization-inspired NNs design DNNs following the optimization algorithms and can be divided into two categories. One to unroll classical optimization or numerical iterative algorithms and introduce learnable parameters, so as to obtain learnable DNNs [21], [22], [23]. The other type replaces one NN's layer with an optimization solver, e.g., [24], [25] consider the solver of quadratic programming (QP) problem as a layer of DNN.

OptEq is relevant but not comparable to the optimization-inspired network. Given the optimization problem, different optimization algorithms may inspire various DNN architectures. However, conversely, given a DNN, it is much harder to identify the underlying optimization problem. Such an *"inverse"* problem is the focus of OptEq. By Lemma 1, we can easily conclude that finding the underlying optimization problem for OptEq is an important step towards understanding general DNNs.

3 The Proposed Optimization Induced Equilibrium Networks

3.1 Preliminaries and Notations

We provide some definitions that are frequently used throughout the paper. A function $f:\mathcal{H}\to\mathbb{R}\cup\{+\infty\}$ is proper if the set $\{x: f(x)<+\infty\}$ is non-empty, where $\mathcal{H}$ is the euclidean space. We write l.s.c for short of lower semi-continuous. The subdifferential, proximal operator and Moreau envelope of a proper convex function f are defined as

$$\begin{cases}\partial f(\mathbf{x}) := \mathbf{g}\in\mathcal{H}: f(\mathbf{y})\geq f(\mathbf{x})+\langle\mathbf{y}-\mathbf{x},\mathbf{g}\rangle, \forall \mathbf{y}\in\mathcal{H},\\ \mathrm{prox}_{\mu\cdot f}(\mathbf{x}) := \mathbf{z}\in\mathcal{H}: \mathbf{z}=\mathrm{argmin}_{\mathbf{u}}\frac{1}{2\mu}\|\mathbf{u}-\mathbf{x}\|^2+f(\mathbf{u}),\\ M_f^{\mu}(\mathbf{x}) := \min_{\mathbf{u}}\frac{1}{2\mu}\|\mathbf{u}-\mathbf{x}\|^2+f(\mathbf{u}),\end{cases}$$

respectively, where $\langle\cdot,\cdot\rangle$ is the inner product and $\|\cdot\|$ is the induced norm. The conjugate f^* of a proper convex function f is defined as: $f^*(\mathbf{y}) := \sup\langle\mathbf{y},\mathbf{x}\rangle - f(\mathbf{x}) : \mathbf{x}\in\mathcal{H}$. For the matrix $\mathbf{W}$, $\|\mathbf{W}\|_2$ is the operator norm. While for the vector ℓ_2-norm, we write $\|\cdot\|$ for simplicity. Given the map $\mathcal{T}:\mathbb{R}^m\to\mathbb{R}^m$, the fixed points set is denoted by $\mathrm{Fix}(\mathcal{T}) = \mathbf{z}\in\mathbb{R}^m : \mathcal{T}(\mathbf{z})=\mathbf{z}$, whose cardinality is $|\mathrm{Fix}(\mathcal{T})|$.

DEQ is inspired by the observation on the feed-forward DNN (with an input-skip connection): for $k=1,\ldots,L-1$,

$$\mathbf{z}_{k+1}=\sigma(\mathbf{W}_k\mathbf{z}_k+\mathbf{U}_k\mathbf{x}+\mathbf{b}_k),\quad \mathbf{y}=\mathbf{W}_{L+1}\mathbf{z}_L, \tag{1}$$

where $\sigma(\cdot)$ is a non-linear activation function, $\mathbf{W}_k\in\mathbb{R}^{n_k\times n_{k-1}}$ and $\mathbf{U}_k\in\mathbb{R}^{n_k\times d}$ are learnable weights and $\mathbf{b}_k\in\mathbb{R}^{n_k}$ is a bias term. A direct way to obtain the equilibrium point of this system is to consider the fixed point equation: $\mathbf{z}^*=\sigma(\mathbf{W}\mathbf{z}^*+\mathbf{U}\mathbf{x}+\mathbf{b})$. And we can utilize any root-finding algorithm to solve this equation. Although DEQ may achieve good performance with a smaller number of parameters than DNNs, its superiority heavily relies on the careful initialization and regularization due to the instability issue of the fixed point problems. Some recent work [8] is devoted to solving the instability issue by using a tricky re-parametrization of the weight matrix $\mathbf{W}$. However, this may greatly weaken the expressive power of DEQ, see Prop. 8 in [9]. Most importantly, in the present equation form, we have difficulty in getting further properties of the equilibrium point.

3.2 One Layer OptEq

Here, we consider an alternative form of the system in Eq. (1).

Lemma 1 (Universal Hidden Unit). *Given the parameters $\{(\mathbf{W}_k,\mathbf{U}_k,\mathbf{b}_k)\}_{k=1}^L$ of a general DNN in Eq. (1), there exists a set of weights $\{\overline{\mathbf{W}}_k\in\mathbb{R}^{n_k\times m}\}_{k=0}^L$ with $m\geq\max\{n_k+n_{k+1}, k=1,\ldots,L-1\}$, such that the system in Eq. (1) can be re-written as the following network: for $k=1,\ldots,L-1$,*

$$\overline{\mathbf{z}}_{k+1}=\overline{\mathbf{W}}_k^{\top}\sigma(\overline{\mathbf{W}}_k\overline{\mathbf{z}}_k+\mathbf{U}_k\mathbf{x}+\mathbf{b}_k),\quad \mathbf{y}=\overline{\mathbf{W}}_{L+1}\overline{\mathbf{z}}_L. \tag{2}$$

Notably, *without changing the output* $\mathbf{y}$, any feed-forward DNN has the reformulation in Eq. (2). The formal proof can be found in appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2022.3181425, we present the main idea here

$$\begin{aligned}\mathbf{y}=&\mathbf{W}_L\sigma(\mathbf{W}_{L-1}\sigma(\cdots\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{z}_0+\mathbf{U}_1\mathbf{x}+\mathbf{b}_1)\cdots))\\ =&\underbrace{\overline{\mathbf{W}}_L\,\overline{\mathbf{W}}_{L-1}^{\top}}_{\mathbf{W}_L}\sigma\left(\underbrace{\overline{\mathbf{W}}_{L-1}\,\overline{\mathbf{W}}_{L-2}^{\top}}_{\mathbf{W}_{L-1}}\sigma\left(\cdots\underbrace{\overline{\mathbf{W}}_2\,\overline{\mathbf{W}}_1^{\top}}_{\mathbf{W}_2}\cdots\right)\right).\end{aligned}$$

In the sense of weight-tying (i.e., all the layers share the same weights), the DNN's output $\mathbf{y}$ is a linear transformation of $\overline{\mathbf{z}}_L$, where $\overline{\mathbf{z}}_L$ is a good approximation of the following fixed point equation under some mild assumptions:

$$\mathbf{z}^*=\mathbf{W}^{\top}\sigma(\mathbf{W}\mathbf{z}^*+\mathbf{U}\mathbf{x}+\mathbf{b}). \tag{3}$$

Hence, the feed-forward DNN also inspires an interesting and different equilibrium model Eq. (3). We call it Optimization Induced Equilibrium Networks (OptEq) since it is tightly associated with an underlying optimization problem. As shown in Theorem 1 that follows, the equilibrium point $\mathbf{z}^*$ is a solution of a convex problem that has an explicit formulation. From the perspective of optimization, we can easily solve the existence and the uniqueness problems of the fixed point equation, rather than resorting to the reparameterization trick. Most importantly, by studying the underlying optimization problem, we can investigate the properties of the equilibrium point of OptEq. The following theorem formally shows the relation between OptEq and optimization.

Assumption 1. *The activation function $\sigma:\mathbb{R}\to\mathbb{R}$ is monotone and $\tilde{L}_{\sigma}$-Lipschitz, i.e.,*

TABLE 1
Examples to Modify the Underlying Optimization Problem

Underlying Convex Function	OptEq Layer	Illustration
$\varphi(\mathbf{z}) = \psi^*(\mathbf{z}) - \frac{1}{2}\|\mathbf{z}\|^2$	$\mathbf{W}^\top\sigma(\mathbf{Wz}+\mathbf{Ux}+\mathbf{b})$	original OptEq
$\alpha M_\varphi^{1-\alpha}(\mathbf{z})$	$\alpha\mathbf{W}^\top\sigma(\mathbf{Wz}+\mathbf{Ux}+\mathbf{b}) + (1-\alpha)\mathbf{z}$	introduce skip connection
$\psi^*(\mathcal{A}(\mathbf{z})) - \frac{1}{2}\|\mathcal{A}(\mathbf{z})\|^2$	$\mathcal{A}^{-1}(\mathbf{W}^\top\sigma(\mathbf{W}\mathcal{A}(\mathbf{z})+\mathbf{Ux}+\mathbf{b}))$	$\mathcal{A}(\cdot)$ is an invariable affine operator, including translation, rotation and scaling
$\mathbf{1}^\top\tilde{\sigma}^*(\mathbf{W}^{-\top}\mathbf{z}) \to \frac{1}{2}\mathbf{z}^\top\mathbf{W}^{-1}\mathbf{S}^{-1}\mathbf{W}^{-\top}\mathbf{z}$	$\mathbf{W}^\top\mathbf{S}(\mathbf{Wz}+\mathbf{Ux}+\mathbf{b})$	for ReLU case in Eq. (4), multivariate activation function
$(\varphi+\gamma\mathcal{R}_z)(\mathbf{z})$	$\mathbf{W}^\top\sigma(\mathbf{W}(\mathbf{z}^*-\gamma\frac{\partial\mathcal{R}_z(\mathbf{z}^*)}{\partial\mathbf{z}})+\mathbf{Ux}+\mathbf{b})$	minimizer of $(\varphi+\gamma\mathcal{R}_z)(\cdot)$, see Theorem 4 for more details

$$0 \le \frac{\sigma(a)-\sigma(b)}{a-b} \le \tilde{L}_\sigma, \quad \forall a,b\in\mathbb{R}, \quad a\neq b.$$

Theorem 1. *If Assumption 1 holds, for one NN layer $f:\mathbb{R}^m\to\mathbb{R}^m$ given by*

$$f(\mathbf{z}) := \frac{1}{\mu}\mathbf{W}^\top\sigma(\mathbf{Wz}+\mathbf{Ux}+\mathbf{b}), \quad \mu \ge \tilde{L}_\sigma\|\mathbf{W}\|_2^2,$$

the solution to the fixed point equation $\mathbf{z}=f(\mathbf{z})$ is the minimizer of the convex function $\varphi(\cdot)$, where,

$$\varphi(\mathbf{z}) = \psi^*(\mathbf{z}) - \frac{1}{2}\|\mathbf{z}\|^2, \ \psi(\mathbf{z}) = \frac{1}{\mu}\mathbf{1}^\top\tilde{\sigma}(\mathbf{Wz}+\mathbf{Ux}+\mathbf{b}),$$

in which $\forall a\in\mathbb{R}, \ \tilde{\sigma}(a)=\int_0^a\sigma(t)\,dt$, applied element-wisely to vectors, and $\mathbf{1}$ is the all one vector. Furthermore, we have $f(\mathbf{z})=\mathrm{prox}_\varphi(\mathbf{z})$, i.e., the NN layer is a proximal operator.

Theorem 1 shows that OptEq's layer given in Eq. (3) is a proximal operator of an underlying convex function given by a conjugate function, and the equilibrium point of OptEq happens to be the minimizer of this function.[1] In the rest part of this paper, for ease of discussion, we focus on the case that $\mu=1$ for OptEq, which may correspond to the assumption $\tilde{L}_\sigma=1$ and $\|\mathbf{W}\|_2\le 1$.

In some cases, we can write down the closed form of the optimization objection $\varphi(\cdot)$. For example, when the weight matrix $\mathbf{W}$ is invertible, and the activation $\sigma(\cdot)$ is ReLU, i.e., $\sigma(x)=\max x,0,\forall x\in\mathbb{R}$, we have

$$\varphi(\mathbf{z}) = \mathbf{1}^\top\tilde{\sigma}^*(\mathbf{W}^{-\top}\mathbf{z}) - \langle\mathbf{Ux}+\mathbf{b},\mathbf{W}^{-\top}\mathbf{z}\rangle - \frac{1}{2}\|\mathbf{z}\|^2, \tag{4}$$

where $\tilde{\sigma}^*(x)=\begin{cases}\frac{1}{2}x^2, & x>0,\\ \infty, & x\le 0\end{cases}$, applied element-wise to vectors. In the ReLU activation case, OptEq is equivalent to solving a QP problem. As a convex optimization layer, QP's powerfulness and effectiveness have been verified in [24], [25].

By Theorem 1, we can quickly obtain the well-posedness of OptEq. In general, any $\mathbf{W}$ that makes the underlying objective φ to be strictly convex will ensure the existence and uniqueness of OptEq's equilibrium. For example, when $\|\mathbf{W}\|_2 < 1$, the operator: $\mathbf{z}\mapsto\mathbf{W}^\top\sigma(\mathbf{Wz}+\mathbf{Ux}+\mathbf{b})$ is contractive, therefore the fixed point equation Eq. (3) has a unique solution, i.e., it exists and is unique. What's more, we show a training strategy that can actually deal with a much more general case — $|\mathrm{Fix}(\mathcal{T})|\ge 1$. Please see Section 5.2 for more details.

OptEq's most attractive aspect is that, by modifying the underlying convex problem, it can inspire many different equilibrium model architectures. For example, we can introduce the commonly used skip connection structure only by replacing $\varphi(\mathbf{z})$ with its Moreau envelope $\alpha M_\varphi^{1-\alpha}(\mathbf{z})$, which does not change the equilibrium point but make OptEq's layer strongly monotone and invertible, and hence can stabilize the iteration. We provide more examples in Table 1. Moreover, we can introduce customized properties of equilibrium point into the model in this way, please see Section. 5.1.

3.3 Deep OptEq

Some work claims that one layer implicit equilibrium is enough [3] and improves the model expressive ability by stacking small DEQs to obtain a wide one-layer DEQ, i.e., considering a fixed point problem in a higher dimension. However, in practice, solving a high-dimensional fixed-point equation is very time-consuming. Therefore, to improve the efficiency of OptEq, we choose to concatenate OptEq and propose deep OptEq, which is a good extension of the one-layer wide one since it improves the model capability without changing the problem scale. Indeed, as we will show in the next section, wide single-layer Opteqs are special cases of deep Opteqs in the asymptotic sense.

In this subsection, we propose a multi-layer version of OptEq, which is also associated to an underlying optimization problem.We consider a multi-layer OptEq, where $\mathbf{x}_0\in\mathbb{R}^{d_x}$ denotes the input, $\mathbf{z}\in\mathbb{R}^m$ denotes the hidden unit, and $\mathbf{y}\in\mathbb{R}^{d_y}$ denotes the output. Namely, deep OptEq follows the implicit equation:

$$\begin{cases}\mathbf{x}=g(\mathbf{x}_0,\mathbf{W}_0),\\ \mathbf{z}=\mathcal{T}(\mathbf{z},\mathbf{x},\boldsymbol{\theta}):=f_L\circ f_{L-1}\cdots\circ f_1(\mathbf{z},\mathbf{x},\boldsymbol{\theta})\\ \mathbf{y}=\mathbf{W}_{L+1}\mathbf{z},\end{cases} \tag{5}$$

where for all $l\in[1,L]$ and $\alpha\in(0,1]$,

$$f_l(\mathbf{z},\mathbf{x}) = \alpha\mathbf{W}_l^\top\sigma(\mathbf{W}_l\mathbf{z}+\mathbf{U}_l\mathbf{x}+\mathbf{b}_l) + (1-\alpha)\mathbf{z}. \tag{6}$$

1. Besides the forms we show, one may expect a common rule to determine whether a general mapping is a proximal operator or not, which can help to create the new equilibrium models from the optimization perspective. We provide the sufficient and necessary conditions in Lemma 5 (see appendix, available in the online supplemental material).

Here, given the set of learnable parameters $\mathbf{W}_0, g(\cdot, \mathbf{W}_0) : \mathbb{R}^{d_x} \to \mathbb{R}^d$ is a continuous function which we usually choose as the feature extractor, e.g., shallow NNs. $\theta = \{(\mathbf{W}_l, \mathbf{U}_l, \mathbf{b}_l)\}_{l=1}^L$ is the set of all learnable parameters for our equilibrium network, where $\mathbf{W}_l \in \mathbb{R}^{n_l \times m}$, $\mathbf{U} \in \mathbb{R}^{n_l \times d}$, $\mathbf{b}_l \in \mathbb{R}^{n_l}$ are the learnable weight matrices and bias term, respectively. Note that $\mathbf{W}_{L+1} \in \mathbb{R}^{d_y \times m}$ is also learnable. $\sigma : \mathbb{R} \to \mathbb{R}$ is the activation function, when the input is multi-dimensional, we apply the function $\sigma(\cdot)$ element-wise. The hidden unit $\mathbf{z}$ is the equilibrium point of the fixed point equation $\mathbf{z} = \mathcal{T}(\mathbf{z}, \mathbf{x}, \theta)$ when $\mathbf{x}$ and θ are given. Without loss of generality, we assume that the feature extractor satisfies a Lipschitz continuity assumption w.r.t. the learnable weight, i.e., $\|g(\mathbf{x}_0, \mathbf{W}_1) - g(\mathbf{x}_0, \mathbf{W}_2)\| \leq L_g \|\mathbf{W}_1 - \mathbf{W}_2\|_2$.

At the first glance, deep OptEq seems very different from the traditional DNNs. Some negative results on DEQ are shown in previous work [9]: with improper weight re-parameterization, DEQ does not contain any feed-forward networks. By contrast, without weight re-parameterization, deep OptEq can include the general feed-forward DNN as its special case.

Lemma 2. *If Assumption 1 holds, the deep OptEqs contain all feed-forward DNNs. More precisely, given a feed-forward DNN in the form*

$$\mathbf{z}_{k+1} = \sigma(\mathbf{A}_k \mathbf{z}_k + \mathbf{c}_k), \quad k = 1, \ldots, L-1, \quad \mathbf{y} = \mathbf{A}_{L+1} \mathbf{z}_L,$$

where $\mathbf{z}_1 = \mathbf{x}$ is the input and $\sigma(\cdot)$ is the activation give by Assumption 1. Then there exists $\{(\mathbf{W}_l, \mathbf{U}_l, \mathbf{b}_l)\}_{l=1}^L$ such that $\mathbf{y}$ is also the output of the corresponding deep OptEq in Eq. (5).

4 Recover Underlying Optimization

This section provides our main results on the connection between convex optimization problem and our deep OptEq. In the previous section, we have shown that one layer of DNN is a proximal operator under a mild assumption. However, the composition of multiple proximal operators is not a proximal operator in most cases. Fortunately, we can still recover the underlying optimization problem of deep OptEq, and find that its equilibrium point is a zero point of a convex function's subdifferential with an additional permutation constraint. In addition, we can explicitly provide the optimization objectives corresponding to deep OptEq in some cases. Before providing the main results, we first show the connection between our deep OptEq given in Eq. (5) and a multi-block one-layer OptEq.

Lemma 3 (Deep OptEq and Multi-block OptEq). *Let $\mathbf{z}_0^*$ be the hidden unit of deep OptEq. Namely $\mathbf{z}_0^*$ is an equilibrium point of the equation $\mathbf{z} = f_L \circ f_{L-1} \cdots \circ f_1(\mathbf{z}, \mathbf{x}, \theta)$, set $\mathbf{z}_1^* := f_1(\mathbf{z}_0^*, \mathbf{x}, \theta)$, $\mathbf{z}_2^* := f_2 \circ f_1(\mathbf{z}_0^*, \mathbf{x}, \theta), \ldots,$ $\mathbf{z}_{L-1}^* := f_{L-1} \cdots \circ f_2 \circ f_1(\mathbf{z}_0^*, \mathbf{x}, \theta)$, then $\widetilde{\mathbf{z}}^* := [\mathbf{z}_1^*, \ldots, \mathbf{z}_{L-1}^*, \mathbf{z}_0^*]^\top \in \mathbb{R}^{mL}$ is an equilibrium point of the equation*

$$\widetilde{\mathbf{z}} = \alpha \widetilde{\mathbf{W}}^\top \sigma\Big(\widetilde{\mathbf{W}} \mathbf{P} \widetilde{\mathbf{z}} + \widetilde{\mathbf{U}} \mathbf{x} + \widetilde{\mathbf{b}}\Big) + (1 - \alpha) \mathbf{P} \widetilde{\mathbf{z}}, \tag{7}$$

where $\widetilde{\mathbf{W}}$ is block diagonal and $\mathbf{P}$ is a permutation matrix,

$$\widetilde{\mathbf{W}} := \begin{bmatrix} \mathbf{W}_1 & & \\ & \ddots & \\ & & \mathbf{W}_L \end{bmatrix}, \quad \mathbf{P} := \begin{bmatrix} 0 & & & \mathbf{I} \\ \mathbf{I} & 0 & & \\ & \ddots & \ddots & \\ & & \mathbf{I} & 0 \end{bmatrix},$$

$\widetilde{\mathbf{U}} := [\mathbf{U}_1, \ldots, \mathbf{U}_L]^\top$ and $\widetilde{\mathbf{b}} := [\mathbf{b}_1, \ldots, \mathbf{b}_L]^\top$ are the concatenated matrix and vector, respectively (see Eq. (15) in appendix for more details), available in the online supplemental material.

By Eq. (7), and the necessary and sufficient condition for one DNN layer to be a proximal-like operator (Lemma 6 in appendix), available in the online supplemental material, we can further reveal the connection between deep OptEq and optimization under a mild assumption.

Assumption 2. *Assumption 1 with $\tilde{L}_\sigma = 1$ and $\|\mathbf{W}_i\|_2 \leq 1, \forall i \in [1, L]$ holds.*

Note that we make this assumption just for the ease of discussion. The assumption is actually unnecessary since we can introduce an additional constant to re-scale the whole operator as we did in Theorem 1.

Theorem 2 (Recovering Optimization Problem from Deep OptEq). *If Assumption 2 holds, then any equilibrium point $\widetilde{\mathbf{z}}^*$ of Eq. (7) satisfies*

$$0 \in \partial \Phi(\widetilde{\mathbf{z}}^*) + (\mathbf{I} - \mathbf{P}) \widetilde{\mathbf{z}}^*, \tag{8}$$

where $\mathbf{I}$ is the identity matrix and $\Phi(\widetilde{\mathbf{z}}^*)$ is given by a sequence Moreau envelopes of convex functions ${\varphi_i}_{i=1}^L$ such that $\operatorname{prox}_{\varphi_i}(\mathbf{z}) = \mathbf{W}_i^\top \sigma(\mathbf{W}_i \mathbf{z} + \mathbf{U}_i \mathbf{x} + \mathbf{b}_i)$, namely

$$\Phi(\widetilde{\mathbf{z}}) = \sum_{i=1}^L \alpha M_{\varphi_i}^{1-\alpha}(\mathbf{z}_i),$$

where $\mathbf{z}_i$ is the ith block of $\widetilde{\mathbf{z}}^*$ and $M_{\varphi_i}^{1-\alpha}(\mathbf{z})$ is the φ_i's Moreau envelope.

When the block size is 1, i.e., $L = 1$, we can immediately obtain that $0 \in \partial \Phi(\widetilde{\mathbf{z}}^*)$, namely, the equilibrium point is a solution of a convex optimization problem. So the results provided in Theorem 1 is a special case here. Note that one block does not mean that $\mathbf{z}$ is one-dimensional. Moreover, for two blocks, the deep OptEq is also an optimization solver.

Corollary 1. *If the block size $L = 2$ and Assumption 2 holds, then the equilibrium point $\widetilde{\mathbf{z}}^* = [\mathbf{z}_1^*, \mathbf{z}_0^*]^\top$ of Eq. (7) is also a solution to a convex problem*

$$\min_{\mathbf{z}_1, \mathbf{z}_0} \left\{ \alpha M_{\varphi_1}^{1-\alpha}(\mathbf{z}_1) + \alpha M_{\varphi_2}^{1-\alpha}(\mathbf{z}_0) + \frac{1}{2} \|\mathbf{z}_1 - \mathbf{z}_0\|^2 \right\}.$$

For general $L > 2$, we can also write down the monotone inclusion equation Eq. (8)'s underlying optimization problem when $\alpha \to 0$. Interestingly, this result implies the equivalence between the composited deep models and the wide shallow ones in the asymptotic sense.

Theorem 3 (Connection between Wide and Deep OptEq). *If Assumption 2 holds, and there is at least one $\|\mathbf{W}_i\|_2 < 1$. Assume $\widetilde{\mathbf{z}}^*(\alpha) := [\mathbf{z}_1^*(\alpha), \ldots, \mathbf{z}_{L-1}^*(\alpha), \mathbf{z}_0^*(\alpha)]^\top \in \mathbb{R}^{mL}$ is the equilibrium point of Eq. (7). When $\alpha \to 0$, all $\mathbf{z}_l^*(\alpha)$s tend to be equal, and the limiting point is $\mathbf{y}$, the last entry of the minimizer $(\mathbf{x}_1, \ldots, \mathbf{x}_L, \mathbf{y})$ of the following optimization problem:*

$$\min_{\mathbf{x}_1, \ldots, \mathbf{x}_L, \mathbf{y}} \left\{ \sum_{l=1}^L \left(\varphi_l(\mathbf{x}_l) + \frac{1}{2} \|\mathbf{x}_l - \mathbf{y}\|^2 \right) \right\},$$

where $\varphi_l(\cdot)$ is the same as that in Theorem 2.

TABLE 2
Some Choices for $\mathcal{R}_z$ and the Corresponding $\mathcal{T}_{\mathcal{R}_z}$

$\mathcal{R}_z$	Operator	Form	Prior Information
$\frac{\|\mathbf{z}\|^2}{2}$	$\operatorname{prox}_{\gamma R_z}$	$\frac{\mathbf{z}}{1+\gamma}$	re-scale, feature decay
$\|\mathbf{z}\|_1$	$\operatorname{prox}_{\gamma R_z}$	$(\lvert\mathbf{z}\rvert-\gamma)_+ \odot \operatorname{sgn}(\mathbf{z})$	shrinkage operator, surrogate of sparsity
$\frac{2}{\|\mathbf{z}\|^2+2\epsilon}$	$\mathcal{I}-\gamma\frac{\partial\mathcal{R}_z}{\partial\mathbf{z}}$	$\left(1-\frac{4\gamma}{(\|\mathbf{z}\|^2+2\epsilon)^2}\right)\mathbf{z}$	feature incay, feature expansion
$\frac{1}{2}\sum_{i\neq j}\left(\frac{\mathbf{z}_i^\top\mathbf{z}_j}{\|\mathbf{z}_i\|\|\mathbf{z}_j\|}\right)^2$	$\mathcal{I}-\gamma\frac{\partial\mathcal{R}_z}{\partial\mathbf{z}_i}$	$\mathbf{z}_i-(\mathcal{I}+\operatorname{Proj}(\mathbf{z}_i))\circ\sum\operatorname{Proj}(\mathbf{z}_{j\neq i})\left(\frac{\mathbf{z}_i}{\|\mathbf{z}_i\|^2}\right)$	feature decorrelation, improve independence, please see Eq. (13), where $\operatorname{Proj}(\mathbf{z}_j):=\frac{\mathbf{z}_j\mathbf{z}_j^\top}{\|\mathbf{z}_j\|^2}$

Theorem 3 implies that, when $\alpha\to 0$, the equilibrium point of the deep OptEq is the same as the solution of $L\mathbf{z}=\sum_{i=1}^{L}\mathbf{W}_i^\top\sigma(\mathbf{W}_i\mathbf{z}+\mathbf{U}_i\mathbf{x}+\mathbf{b}_i)$, which is a wide one-layer OptEq with multiple blocks. Given the output dimension and the same amount of learnable parameters, the wide multi-block OptEq is actually a special case of deep OptEq in the asymptotic sense. Hence, deep OptEq is more expressive than wide one-layer OptEq.

In general, we can still loosely treat the equilibrium point as a minimizer of an implicit optimization problem, since the only difference between the monotone inclusion equation $0\in\partial\Phi(\widetilde{\mathbf{z}}^*)$ and Eq. (8) is an additional constraint dominated by the operator $(\mathbf{I}-\mathbf{P})(\cdot)$, which aims to reduce the divergence between the multi-blocks. For example, when $L=2$, the results in Corollary 1 show that we need to simultaneously consider the sum of convex objective and the distance $\|\mathbf{z}_1-\mathbf{z}_0\|$.

5 Introducing Customized Properties

By employing the underlying optimization problem, we can investigate the potential property of the equilibrium points. A more advanced way to use the connection between (deep) OptEq and optimization is to introduce some customized properties to equilibrium points, i.e., the feature learned by OptEq. Note that none of the previous DEQs take into account the regularization of features, which has been proved to be effective both theoretically [26], [27] and empirically [28].

5.1 Underlying Optimization Inspired Feature Regularization

As we mentioned in Section 3.2, if we replace the underlying optimization objective with its Moreau envelope, OptEq will naturally have a skip-connection structure, which has been adopted in the construction of deep OptEq. Following this idea, when we modify the underlying optimization problem of deep OptEq, it should inspire more network architectures.

An exciting application of Theorem 2 is introducing customized properties by modifying $\Phi(\cdot)$: appending one layer after deep OptEq is equivalent to adding one term to the objective $\Phi(\cdot)$. Specifically, we have the following theorem.

Theorem 4. *With the same setting as in Theorem 1. The fixed point of the equation $\mathbf{z}^*=f\circ(\mathcal{I}-\gamma\frac{\partial\mathcal{R}_z}{\partial\mathbf{z}})(\mathbf{z}^*)$, namely*

$$\mathbf{z}^*=\mathbf{W}^\top\sigma\left(\mathbf{W}\left(\mathbf{z}^*-\gamma\frac{\partial\mathcal{R}_z(\mathbf{z}^*)}{\partial\mathbf{z}}\right)+\mathbf{U}\mathbf{x}+\mathbf{b}\right),$$

is the minimizer of the convex function $(\varphi+\gamma\mathcal{R}_z)(\cdot)$.

Hence, if We modify $\Phi(\widetilde{\mathbf{z}})$ to $\Phi(\widetilde{\mathbf{z}})+\mathcal{R}_z(\mathbf{z}_L)$, then deep OptEq becomes

$$\mathbf{z}=\mathcal{T}(\mathcal{T}_{\mathcal{R}_z}(\mathbf{z}),\mathbf{x},\boldsymbol{\theta}),$$

where $\mathcal{T}_{\mathcal{R}_z}=\operatorname{prox}_{\gamma R_z}$ or $\mathcal{T}_{\mathcal{R}_z}=\mathcal{I}-\gamma\frac{\partial\mathcal{R}_z}{\partial\mathbf{z}}$ when the proximal is hard to calculate. For the choice of $\mathcal{R}_z$, we give several examples in Table 2.

In general, $\mathcal{R}_z(\cdot)$ can be any convex function that contains the prior information of the feature. In summary, we introduce feature regularization by modifying the underlying optimization problem, which leads to a change of network structure. Once again, studying the equilibrium models from the perspective of optimization shows great advantages.

5.2 SAM Iteration Induced Feature Regularization

In this subsection, we provide another strategy for feature regularization. Note that most previous DEQs are devoted to ensuring a singleton fixed point set, relying on the tricky weight re-parameterization. Considering the general case — $|\operatorname{Fix}(\mathcal{T})|\geq 1$, we can choose the equilibrium with desired property by solving the following constrained optimization problem:

$$\mathbf{z}^*(\mathbf{x},\boldsymbol{\theta}):=\underset{\mathbf{z}\in\operatorname{Fix}(\mathcal{T}(\cdot,\mathbf{x},\boldsymbol{\theta}))}{\operatorname{argmin}}\ \mathcal{R}_z(\mathbf{z}),\tag{9}$$

where $\mathcal{R}_z(\cdot)$ is the feature regularization that contains the prior information of the feature. Given the training data $(\mathbf{x}_0,\mathbf{y}_0)\in\mathbb{R}^{d_x}\times\mathbb{R}^{d_y}$, the whole training procedure becomes[2]

$$\min_{\widetilde{\boldsymbol{\theta}}}\ell(\mathbf{W}_{L+1}\cdot\mathbf{z}^*(\mathbf{x},\ \boldsymbol{\theta}),\mathbf{y}_0)+\mathcal{R}_w(\widetilde{\boldsymbol{\theta}}),\tag{10}$$

where $\widetilde{\boldsymbol{\theta}}:=\mathbf{W}_0,\boldsymbol{\theta},\mathbf{W}_{L+1}$, $\mathbf{x}$ is given by Eq. (5), $\mathcal{R}_w(\cdot)$ is the regularizer on the parameters, e.g., weight decay, and $\ell:\mathbb{R}^{d_y}\times\mathbb{R}^{d_y}\to\mathbb{R}^+$ is the loss function.

2. For the sake of clarity, we utilize one training pair for discussion. In general, all the discussed results hold when we replace the single data point with the whole data set.

We adopt the Sequential Averaging Method (SAM) [10] to solve the problem Eq. (9). Starting from any $\mathbf{z}_0 \in \mathbb{R}^m$, we consider the following sequence $\mathbf{z}^k{}_{k\in\mathbb{N}}$:

$$\mathbf{z}^k = \beta_k \mathcal{S}_{\lambda_k}(\mathbf{z}^{k-1}) + (1-\beta_k)\mathcal{T}(\mathbf{z}^{k-1}, \mathbf{x}, \boldsymbol{\theta}), \tag{11}$$

where $\{\beta_k\}_{k\in\mathbb{N}}$ and $\{\lambda_k\}_{k\in\mathbb{N}}$ are sequences of real numbers in (0,1], $\mathcal{S}_\lambda(\mathbf{z}) = (1-\gamma\lambda)\mathbf{z} - \gamma\frac{\partial \mathcal{R}_z(\mathbf{z})}{\partial \mathbf{z}}$, where $\gamma \in [0,1]$ is a hyper-parameter. Then we choose the Kth iteration $\mathbf{z}^K(\mathbf{x},\boldsymbol{\theta})$ as an approximate of $\mathbf{z}^*(\mathbf{x},\boldsymbol{\theta})$ and put it in the final loss term

$$\min_{\widetilde{\boldsymbol{\theta}}} \ell\big(\mathbf{W}_{L+1}\cdot \mathbf{z}^K(\mathbf{x},\ \boldsymbol{\theta}), \mathbf{y}_0\big) + \mathcal{R}_w(\widetilde{\boldsymbol{\theta}}). \tag{12}$$

The unrolling term $\mathbf{z}^K(\mathbf{x},\ \boldsymbol{\theta})$ aggregates information from both $\mathcal{R}_z(\mathbf{z})$ and $\mathcal{T}$, making the prior information of feature being an inductive bias during training. And our model can be easily trained by any first-order optimization algorithms, e.g., GD, SGD, Adam, etc.

Remark 1. SAM needs $\mathcal{R}_z(\cdot)$ to be strongly convex, here we use a modified version of SAM which only assumes convexity. Given well-chosen $\{\beta_k\}_{k\in\mathbb{N}}$ and $\{\lambda_k\}_{k\in\mathbb{N}}$, we can prove that the sequence generated by Eq. (11) converges to the point $\mathbf{z}^*(\mathbf{x},\boldsymbol{\theta})$. Furthermore, we prove that the whole training dynamic, using the unrolling SAM strategy with backpropagation (BP), converges with a linear convergence rate.

Remark 2. If we use the method in Section 5.1 to introduce the prior information, there is no need to use SAM iteration again. Therefore, we can let beta $\beta_k = 0$, $\mathcal{S}_{\lambda_k}(\cdot) = 0$, and use the unrolling fixed point iteration strategy during training. Note that we can also use the implicit function theorem (IFT) based training way.

Previous equilibrium models utilize IFT in training to avoid the storage consumption of forward-propagation. The cost for that is it needs to solve two large-scale linear equations (or perform the matrix inversion directly) during training. Deep OptEqs can be trained both in the unrolling based and IFT based ways. In the sense of BP, the two training ways have different merits and limitations. We provide comparative experiments in Section 7 and detailed discussion in the following subsection.

5.3 Discussion About the Training Methods

The training method is summarised in Algorithm 1.

The IFT based implicit way utilizes the limited memory to train the model and is insensitive to the equilibrium point finding algorithms. However, it consumes much computation budget to solve the equation during the inference and BP. On the other hand, the way that unrolls the fixed point finding method may induce implicit bias [29], [30] and consumes much memory during training, but it is faster to infer and train. Note that implicit bias is a double-edged sword; The proposed SAM method can aggregate the information from the prior regularization and the fixed point equation. Hence, the implicit bias becomes a controllable inductive bias.

The tradeoff between memory and computing efficiency for the implicit and unrolling training methods is quite common in the other learning community, such as meta-learning [31] and hyper-parameter optimization [19]. Similarly, for DEQ, the two training ways are neither good nor bad. We should choose them in proper circumstances.

Algorithm 1. Training Algorithm for (Deep) OptEq

```
Input: training data $(\mathbf{x}_{0,i}, \mathbf{y}_i)_{i=1}^n$, unrolling number $K$, iteration number $T$, feature extractor $g(\cdot)$
1:  for $t = 0$ to $T$ do
2:    forward: $\mathbf{x} = g(\mathbf{x}_0, \mathbf{W}_0)$, set $\mathbf{z}_0 = \mathbf{x}$
3:      if IFT-based training then
4:        solve the fixed point equation $\mathbf{z}^* = \mathcal{T}(\mathcal{T}_{\mathcal{R}_z}(\mathbf{z}^*), \mathbf{x})$;
5:      else if unrolling with SAM then
6:        for $k = 1$ to $K$ do
7:          $\mathbf{z}^k = \beta_k \mathcal{S}_{\lambda_k}(\mathbf{z}^{k-1}) + (1-\beta_k)\mathcal{T}(\mathbf{z}^{k-1}, \mathbf{x})$;
            /* SAM by Eq. (11) */
8:        end
9:        $\mathbf{z}^* = \mathbf{z}^K$;
10:     else /* raw unrolling */
11:       for $k = 1$ to $K$ do
12:         $\mathbf{z}^k = \mathcal{T}(\mathcal{T}_{\mathcal{R}_z}(\mathbf{z}^{k-1}), \mathbf{x})$;
13:       end
14:       $\mathbf{z}^* = \mathbf{z}^K$;
15:     end
16:   backward:
17:     evaluate the loss $L := \ell(\mathbf{W}\mathbf{z}^*, \mathbf{y}) + \mathcal{R}_w(\widetilde{\boldsymbol{\theta}})$;
18:     if unrolling-based training then
19:       get $\partial L/\partial\widetilde{\boldsymbol{\theta}}$ by automatic differentiation;
20:     else if IFT-based training then
21:       get $\partial L/\partial\widetilde{\boldsymbol{\theta}}$ by implicit differentiation;
22:     update the learnable parameters $\widetilde{\boldsymbol{\theta}}$;
23: end
```

6 Convergence Analysis

This section offers the convergence results: (i) the sequence generated by Eq. (11) converges to some point $\mathbf{z}^* \in \mathrm{Fix}(\mathcal{T})$ such that $\mathcal{R}_z(\mathbf{z}^*) \le \mathcal{R}_z(\mathbf{z}),\ \forall \mathbf{z} \in \mathrm{Fix}(\mathcal{T})$; (ii) gradient descent can find a global minimum for the model in Eq. (12).

6.1 Approximation of Equilibrium Point

Note that we approximate the points $\mathbf{z}^*(\mathbf{x},\boldsymbol{\theta})$ by the iterative steps in Eq. (11). In fact, we take the iterative step by extending an existing algorithm, SAM, which was developed in [10] for solving a certain class of fixed-point problems, and then was applied to the bilevel optimization problems [11]. However, the existing SAM method can only deal with strongly convex $\mathcal{R}_z(\mathbf{z})$. Our method is the first SAM type algorithm that can solve the general convex problem restricted to a nonexpansive operator's fixed point set. The following theorem provides the formal statement and the required conditions. Since during the forward-propagation, $(\boldsymbol{\theta}, \mathbf{x})$ is fixed, for the sake of convenience, we simplify $\mathcal{T}(\mathbf{z}, \mathbf{x}, \boldsymbol{\theta})$ as $\mathcal{T}(\mathbf{z})$.

Theorem 5 (Convergence of Modified SAM Iterates). *Suppose that $\nabla\mathcal{R}_z(\mathbf{z})$ is L_z-Lipschitz, and that for any $\beta \in [0, \frac{1}{2}], \lambda \in [0, \frac{L_z}{2}]$, the fixed point set of equation: $\mathbf{z} = \beta(\mathbf{z} - \gamma(\nabla\mathcal{R}_z(\mathbf{z}) + \lambda\mathbf{z})) + (1-\beta)\mathcal{T}(\mathbf{z})$ is uniformly bounded by B_1^* (in norm $\|\cdot\|$) w.r.t. β and λ. Suppose that convex function $\mathcal{R}_z(\mathbf{z})$ has a unique minimizer $\bar{\mathbf{z}}$ on $\mathrm{Fix}(\mathcal{T})$. Let $\beta_k = \frac{\eta}{k^\rho}, \lambda_k = \frac{\eta}{k^c}, \gamma = \frac{1}{2L_z}$, where $\rho, c > 0$, $\rho + 2c < 1$ and $\eta =$*

$\min\{\sqrt{2L_z}, \frac{L_z}{2}, \frac{1}{2}\}$, *then the sequence* $\{\mathbf{z}^k\}_{k\in\mathbb{N}^+}$ *generated by Eq. (11) converge to* $\bar{\mathbf{z}}$.

The formal assumptions of Theorem 5 seem complicated, however, they can be easily fulfilled when $\mathcal{T}(\mathbf{z})$ is contractive. A sufficient condition that makes $\mathcal{T}(\mathbf{z})$ contractive is to let one $\|\mathbf{W}_i\|_2 \leq \zeta < 1$. More specifically, if some $\|\mathbf{W}_i\|_2 \leq \zeta < 1$, then $\mathbf{z} \mapsto \beta(\mathbf{z} - \gamma(\nabla\mathcal{R}_z(\mathbf{z}) + \lambda\mathbf{z})) + (1-\beta)\mathcal{T}(\mathbf{z})$ is contractive and has a unique fixed point, which depends continuously on β and λ [32], and thereby have a uniform bound.

6.2 Global Convergence of Implicit Model

Most previous works on DEQs lack the convergence guarantees for their training. However, analyzing the learnable parameters' dynamics is crucial since it may weaken many model constraints and greatly broaden the function class that the implicit model can represent. For example, the one-layer DEQ, given in [8], maintains the positive definiteness of $(\mathbf{I} - \mathbf{W})$ for all weight $\mathbf{W}$ in $\mathbb{R}^{m\times m}$ through a complicated parameterization technique. However, after analysis, we find that the learnable weight will stay in a small compact set during training, thus, we may only need the positive definiteness within a local region instead of global space for the DEQ [8].

Theorem 6 (Global Convergence (informal)). *Suppose that the initialized weight* $\mathbf{W}_l$*'s singular values are lower bounded away from zero for all* $l \in [1, L+1]$*, and the fixed point set* $\mathrm{Fix}(\mathcal{T}(\cdot, \mathbf{X}, \boldsymbol{\theta}))$ *is non-empty and uniformly bounded for any* $\widetilde{\boldsymbol{\theta}}$ *in a pre-defined compact set. Assume that the activation function is Lipschitz smooth, strongly monotone and 1-Lipschitz. Define constants* Q_0*,* Q_1 *and* Q_3*, which depend on the bounds for initialization parameters, initial loss value, and the datasize. Let the learning rate be* $\eta < \min\{\frac{1}{Q_0}, \frac{1}{Q_1}\}$*. If for all* $l \in [1, L]$*, we have* $n_l = \Omega(poly(N))$*, then the training loss vanishes at a linear rate as*

$$\ell(\widetilde{\boldsymbol{\theta}}^t) \leq \ell(\widetilde{\boldsymbol{\theta}}^0)(1 - \eta Q_0)^t,$$

where t is the number of iteration. Furthermore, the network parameters also converge to a global minimizer $\widetilde{\boldsymbol{\theta}}^*$ *at a linear speed*

$$\|\widetilde{\boldsymbol{\theta}}^t - \widetilde{\boldsymbol{\theta}}^*\| \leq Q_3(1 - \eta Q_0)^{t/2}.$$

Theorem 6 shows that GD converges to a global optimum for any initialization satisfying the boundedness assumption. In general, the lower bounded assumption on singular values is easy to fulfill. With high probability, the weight matrix's singular values are lower bounded away from zero when it is a rectangle and has independent, sub-Gaussian rows or has independent Gaussian entries, see Thm.4.6.1 and Ex.7.3.4 in [33].

6.2.1 About Boundness and Well-Posedness

Similar to the remark after Theorem 5, the existence and boundedness assumption on the set $\mathrm{Fix}(\mathcal{T})$ is mild. Suppose that there exists $l \in [1, L]$, such that $\mathbf{W}_l \leq \zeta < 1$, then the fixed point of $\mathcal{T}$ exists and is unique, and is continuous w.r.t. the parameters $\widetilde{\theta}$. Moreover, via over-parameterization, a proper Gaussian initialization followed by gradient descent produces a sequence of iterates that stay inside a small perturbation region centered at the initial weights. In such a small perturbation region, the largest singular value of each weight $\mathbf{W}_l$ does not change a lot, namely, $\mathbf{W}_l \leq 1$ holds during training. Therefore, throughout the training process, the fixed points are uniformly upper bounded, and the fixed point of $\mathcal{T}$ exists and is unique.

7 Experiments

In this section, we investigate the empirical performance of deep OptEq from three aspects. First, on the image classification problem, we evaluate the performance of deep OptEqs along with our feature regularization strategies. The results trained with different αs are also reported. Second, we compare deep OptEqs with previous implicit models and traditional DNNs. Finally, we compare our unrolling-based method with the IFT-based method and investigate the influence of unrolling iteration number K. Furthermore, we present the results on Cityscapes for semantic segmentation.

Training Strategy of Deep OptEqs. In order to compare the effect of feature regularization on performance in detail, we compare three unrolling training ways based on Eq. (12): (1) $\mathcal{R}_z^{\dagger}$: strategy in Section 5.2, using the proposed SAM given in Eq. (11); (2) $\mathcal{R}_z^{*}$: strategy in Section 5.1, and set $\beta_k = 0$ in Eq. (11); and (3) no Reg: without feature regularization. Here we set the iterative number $K = 20$ for the experiments on CIFAR-10 and let $K = 5$ for the experiments on ImageNet. We discuss the influence of different K in Section 7.4.

7.1 Effects of Different Regularizers

We construct the deep OptEqs with 5 convolutional layers, using five 3×3 convolution kernels with the numbers of channels being $16, 32, 64, 128, 128$. During backward propagation, we utilize the commonly used first order optimization algorithm — SGD. We set the learning rate as 0.1 at the beginning and halve it after every 30 epochs. And the total training epoch is 200.

In this experiment, we compare the performance of two ways to introduce the feature regularization on the CIFAR-10 dataset. We adopt the feature decorrelation as the $\mathcal{R}_z$ here. We also use two norm regularizations to show whether there is a corresponding effect on the learned feature of deep OptEq. Moreover, we show how the hyperparameter α affects the model performance.

The results are shown in Table 3. With the same size of parameters, deep OptEqs beats the general DNN (given in Eq. (1)) easily. It turns out that there is no linear relationship between the performance and the hyperparameter α. The hyperparameter α serves as a trade-off between the effect of fixed point equation and the regularization induced by operator $\mathcal{S}$. In our setting, with a small initialization for $\{\mathbf{W}_l\}$s, all weights will stay in a small compact set during training (see proof of Theorem 6). Therefore when α approaches 1, deep OptEq is an intense contraction (i.e., with a small contractive coefficient), and the SAM iterations will quickly converge to the fixed point, in which case regularization induced operator $\mathcal{S}$ has a limited impact. When α approaches 0, deep OptEq

TABLE 3
(a) The Testing Accuracy (Acc.) of Deep OptEq With Different Settings

α	0.01	0.1	0.4	0.8	1.0
Acc-(no Reg)	58.4%	56.8%	86.9%	**87.4**%	87.2%
Acc-($\mathcal{R}_z^\dagger$)	72.7%	61.5%	86.5%	87.0%	**87.7%**
Acc-($\mathcal{R}_z^*$)	72.6%	60.0%	**88.0**%	87.6%	87.5%
Acc-DNN.			82.7%		
$\mathcal{R}_z^* = \lambda\|\cdot\|_1$	0.01		0.15		0.5
mean $\|\cdot\|_1$	> 5		0.81		**0.34**
Acc.	86.4%		**87.7**%		86.9%
$\mathcal{R}_z^* = \lambda\|\cdot\|^2$	0.01		1.0		10.0
mean $\|\cdot\|^2$	> 10		1.56		**0.82**
Acc.	87.3%		**87.6**%		86.7%

$\mathcal{R}_z^*$ and $\mathcal{R}_z^\dagger$ represent the regularization given in Section 5.1 and Section 5.2, respectively. We set different λ for $\mathcal{R}_z^*$ and the mean values are taken on the whole feature tensor. The total number of parameters is 199 k.

is to be more like the identity operator, so $\mathcal{S}$ dominates the whole iterations.

The optimization inspired implicit regularization ($\mathcal{R}_z^*$) is also an efficient feature regularization method since it modifies deep OptEq structure directly. Here we present two $\mathcal{R}_z^*(\cdot)$ candidates: $\lambda\|\cdot\|_1$ and $\lambda\|\cdot\|^2$. The outputs of the feature show decreases in the corresponding norm, and a suitable regularization coefficient can lead to a better performance.

7.2 Performance of Different Feature Regularizers

On the dataset CIFAR-10, the experiment in this subsection detailedly shows the effect of different settings on regularizer $\mathcal{R}_z^\dagger$ (for Section 5.2), regularizer $\mathcal{R}_z^*$ (for Section 5.1), and α. Note that in this paper, we mainly focus on feature regularizers. However, another line of work that considers some special (weight) regularization techniques for DEQ has also attracted some attention. For example, [34] suggests a regularization method to stabilize DEQ training by explicitly regularizing the Jacobian of the fixed-point iteration equations. [35] shows that state-dependent inexact gradient brings additional training stability regarding the Jacobian spectral radius, which can be understood as a kind of implicit Jacobian regularization. The reader can refer to [34], [35] and the reference therein for more details. We first introduce a regularizer—Hilbert-Schmidt Independence Criterion (HSIC), which is a feature disentanglement method.

7.2.1 HSIC

HSIC is a statistical method to test independence. Compared with the decorrelation method we will present in the following, HSIC can better capture the nonlinear dependency between random variables. We apply HSIC to the feature space. Many works [36], [37] show that when the features learned by the network are uncorrelated, the model usually obtains a good generalization performance. For any pair of random variables $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_B), \mathbf{Y} = (\mathbf{y}_1, \ldots, \mathbf{y}_B)$, where B is the batch size, we utilize the biased finite-sample estimator of HSIC [38]

$$\mathrm{HSIC}(\mathbf{X}, \mathbf{Y}) := (B-1)^{-2}\ \mathrm{tr}(\mathbf{K}_X \mathbf{H} \mathbf{K}_Y \mathbf{H}),$$

where $\mathbf{K}_X$ and $\mathbf{K}_Y$ are the kernel matrices w.r.t. Gaussian RBF kernel of $\mathbf{X}$ and $\mathbf{Y}$, and $\mathbf{H}$ is the centering matrix $\mathbf{H} = \mathbf{I} - B^{-1}\mathbf{1}_B\mathbf{1}_B \in \mathbf{R}^{B\times B}$. Following [28], we aim to eliminate all the correlations between feature maps. To this end, our HSIC regularization is

$$\mathcal{R}_z(\mathbf{Z}) = \sum_{1\le i<j\le m} \mathrm{HSIC}(\mathbf{Z}_{i,:}, \mathbf{Z}_{j,:}).$$

Note that HSIC is a nonparametric regularization term, so it does not increase the parameter size of deep OptEq. The computing cost of HSIC grows as the batch size and feature dimension increase. Some tricks, such as Random Fourier Features approximation [28], can be applied to speed up the calculation. In addition, Theorem 5 is only guaranteed for convex regularization, while HSIC regularization is non-convex. In this paper, we report the great empirical superiority of HSIC and leave the above issues to future work.

7.2.2 Settings and Results

Other Regularizers. Here is the function we utilize to introduce the customized property to the equilibrium point, see Section 5 for more details. For the regularizer $\mathcal{R}_z(\cdot)$ (both for $\mathcal{R}_z^\dagger$ and $\mathcal{R}_z^*$), we set four different settings: (1) $\mathcal{R}_z(\mathbf{Z}) = \sum_{1\le i<j\le m} \mathrm{HSIC}(\mathbf{Z}_{i,:}, \mathbf{Z}_{j,:})$; (2) $\mathcal{R}_z(\mathbf{z}) = \frac{1}{2}\|\mathbf{z}\|^2$; (3) $\mathcal{R}_z(\mathbf{z}) = 1/(\|\mathbf{z}\|^2 + \epsilon)$ which is explored in [39]; (4) Decorrelation: for the B-batch equilibrium points matrix $\mathbf{Z} \in \mathbb{R}^{m\times B}$:

$$\mathcal{R}_z(\mathbf{Z}) = \frac{1}{2}\left\|\mathbf{D}\mathbf{Z}\mathbf{Z}^\top\mathbf{D} - \mathbf{I}\right\|_F^2 := \mathcal{F}_{Dz}(\mathbf{Z}), \tag{13}$$

where $\mathbf{D}$ is a diagonal matrix whose non-zero entries are $\frac{1}{\|\mathbf{Z}_{i,:}\|}$ and $\mathbf{Z}_{i,:}$ is the ith row of the matrix $\mathbf{Z}$. Note that $\mathcal{R}_z(\mathbf{Z})$ here aims at reducing redundant information between feature dimensions, which has been discussed in [40].

Settings. In this experiment, we set $K = 20$ and utilize weight decay to regularize the learnable parameters, i.e., $\mathcal{R}_w(\cdot) = \xi\|\cdot\|^2$, where we choose $\xi = 3e-4$. We utilize the commonly used SGD to train the model. We set the learning rate as 0.1 at the beginning and decay it by 0.7 after every 20 epochs. And the total training epoch is 200. The batch size is 125 in this experiment. We construct the deep OptEqs with 5 convolutional layers, using five 3×3 convolution kernels with the numbers of channels being $16, 32, 64, 128, 128$. The total number of learnable parameters is 199 k.

Results. The results with different regularizers are presented in Table 5. We can see that either adopting SAM iteration or changing the underlying convex optimization problem both improves classification performance. Note that, given the same type of regularization, modifying the underlying optimization problem, i.e., using $\mathcal{R}_z^*$, usually make more improvements. Indeed, to modify the underlying optimization problem, we need to change the architecture of deep OptEq, which has a more direct impact on the model than turning the training loss by $\mathcal{R}_z^\dagger$. When $\alpha = 0.01$, deep OptEq is almost equivalent to the one-layer wide OptEq (see Theorem 3), which is far outperformed by deep OptEq for $\alpha > 0.1$. Compared with other results, $\alpha = 0.1$ gives a poor result, which implies that the performance is

TABLE 4
Comparisons With Previous Implicit Models

Methods	Reg. or Settings	# params	Acc.
Deep OptEqs	Decorrelation ($\mathcal{R}^*\mathbf{z}$)	1.4 M	91.0%
	Decorrelation ($\mathcal{R}^\dagger\mathbf{z}$)	162 k	86.0%
	HSIC ($\mathcal{R}^*\mathbf{z}$)	162 k	87.4%
	No Reg	162 k	85.7%
ODEs	Neural ODE	172 K	53.7%
	Aug. Neural ODE	172 k	60.6%
MONs	Single conv	172 K	74.1%
	Single conv (large)	854 K	82.5%

not monotonic to parameter α. Fortunately, from the table, setting $\alpha > 0.4$ is a safe choice. We notice that the overall performance of feature disentanglement methods (decorrelation and HSIC) are better than the other types of regularization terms whether we utilize it as $\mathcal{R}_z^\dagger$ or $\mathcal{R}_z^*$.

7.3 Comparison With Previous Implicit Models

In this experiment, we compare deep OptEq with other implicit models MDEQ [4], NODEs [1], Augmented NODEs [41], single convolutional Monotone DEQs [8] (short as MON), and classical ResNet-xx [2]. Note that deep OptEqs do not require the additional re-parameterization like MONs [8]. Therefore, our OptDeq models cannot guarantee the uniqueness of the fixed point. However, the proposed SAM training strategy can select the point with the minimal regularization value when the fixed point set in not a singleton.

7.3.1 Results on CIFAR-10

For fair comparisons, we construct the deep OptEqs with 5 convolutional layers. In order to construct deep OptEqs with a similar number of parameters as baseline methods, we use five 3×3 convolution kernels with the numbers of channels being $16, 32, 64, 64, 128$. Moreover, we only use a single convolutional layer as the feature extractor $g(\cdot)$ for the model with 162 k parameters, which is the same as single convolution MONs [8].

The results are shown in Table 4. Notably, even *without feature regularization trick*, our deep OptEqs significantly outperform baseline methods. We highlight the performance of deep OptEqs on CIFAR-10 which outperforms Augmented Neural ODE by 25.1% and MON by 11.6% with *fewer parameters*. Without adding the number of parameters, feature regularization helps deep OptEq to achieve better performance easily. Notably, HSIC, a feature disentanglement regularization, provides a significant gain for the generalization.

7.3.2 Results on ImageNet

We now consider the ability of OptEq on a large-scale dataset with higher-resolution images—ImageNet [42].

Extractors. To train OptEq on ImageNet, we choose a slightly more complex extractor $g(\cdot)$. In this experiment, we test deep OptEq on two extractors: (1) the first two stages of ResNet-50, whose model size is 8 M, and maps the images from the size $224 \times 224 \times 3$ to a feature map belongs to $\mathbb{R}^{28\times 28\times 512}$; (2) the first two stages of Wide-ResNet-50 [43], with model size 24 M and also maps the image to the size of $\mathbb{R}^{28\times 28\times 512}$. Note that the parameters in these extractors are also trained from scratch. We *do not* utilize any pre-trained weights here. We also report the vanilla results on these extractors, i.e., directly append the classification layer after the extractors (refer as Extr. + Cls.).

Architecture. Given the feature map $\mathbf{z} \in \mathbb{R}^{c_{in}\times w\times h}$, different from other experiments in this paper, which performs the linear transformation by a general convolution operator, namely, $\mathbf{W}_l\mathbf{z} := \mathrm{conv}(\mathbf{z}, \mathbf{W}_l)$, where $\mathbf{W}_l \in \mathbb{R}^{c_{out}\times c_{in}\times 3\times 3}$ is the convolutional kernel. For OptEqs on ImageNet, we choose the depthwise separable convolutions [44], i.e., $\mathbf{W}_l\mathbf{z} := \mathrm{conv}(\mathrm{conv}(\mathbf{z}, \mathbf{W}_{l,1}), \mathbf{W}_{l,2}))$, where $\mathbf{W}_{l,1} \in \mathbb{R}^{c_{in}\times 1\times 3\times 3}$ and $\mathbf{W}_{l,2} \in \mathbb{R}^{c_{out}\times c_{in}\times 1\times 1}$ are the convolutional kernels. The architecture details are summarized in Table 6.

Settings. We choose the the decorrelation function Eq. (13) as the regularizer here and adopt the regularizer setting given in Section 5.1. For computational and memory efficiency, we let $K = 5$ here. The optimizer in this experiment is AdamW with weight decay being 0.05, learning rate being 0.001 with cosine decay scheduler and batch size being 1024. We train OptEqs for 300 epochs.

Results. Table 7 shows the accuracy of two different size deep OptEqs, i.e., OptEq-small and OptEq-Mid, in comparison to well-known reference models in computer vision. OptEqs outperforms the current SOTA equilibrium models and are remarkably competitive with some strong explicit models. For example, the OptEq-small with 18 M parameters outperforms DEQ (classical equilibrium models with 18 M parameters), ResNet-34 (21 M parameters), and even ResNet-50 (26 M parameters). The larger OptEq-Mid (40 M parameters) reaches higher level of performance comparing to ResNet-101 (52 M parameters) and MDEQ-large (SOTA equilibrium models with 63 M parameters). OptEq's results are far beyond the scale and accuracy levels of prior equilibrium models.

TABLE 5
The Testing Accuracy of Deep OptEq With Different Settings

α	No Reg	$\left(\frac{\|\cdot\|^2}{2}\right)^\dagger$	$\left(\frac{2}{\|\cdot\|^2+2\epsilon}\right)^\dagger$	$\left(\frac{2}{\|\cdot\|^2+2\epsilon}\right)^*$	$\mathcal{F}^\dagger_{Dz}$	$\mathcal{F}^*_{Dz}$	HSIC†	HSIC*
0.01	58.4%	69.4%	**75.0%**	64.9%	72.7%	72.6%	70.6%	72.2%
0.1	56.8%	61.9%	63.2%	64.7%	61.5%	60.0%	**66.1**%	64.5%
0.4	86.9%	87.3%	78.4%	87.7%	86.5%	**88.0%**	85.1%	85.5%
0.8	87.4%	87.3%	87.3%	87.5%	87.0%	**87.6%**	**87.6%**	87.5%
1.0	87.2%	87.4%	87.0%	87.6%	87.7%	87.5%	**88.1%**	87.9%

We denote by "no Reg" the SAM with $\beta_k = 0$. And the scripts † and $*$ means the regularizer given in Section 5.2 and Section 5.1, respectively.

TABLE 6
The Detailed Archit. of OptEqs

	Extractor (size)	Archit. of OptEq	Archit. of cls. layer	Model Size
OptEq-Small	ResNet-50 (8 M)	$\begin{bmatrix} 3\times 3\times 1, 512 \\ 1\times 1\times 512, 512 \end{bmatrix} \times 5$	$\begin{bmatrix} 1\times 1\times c_{in}, c_{out} \\ \text{ReLU} \\ \text{MaxPool} \end{bmatrix} \times 2$ + AvgPool	18 M (8M+7M+3 M)
OptEq-Midddle	Wide-ResNet-50 (24 M)	$\begin{bmatrix} 5\times 5\times 1, 1024 \\ 1\times 1\times 1024, 512 \\ 3\times 3\times 1, 512 \\ 1\times 1\times 512, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times 1\times c_{in}, c_{out} \\ \text{ReLU} \\ \text{MaxPool} \end{bmatrix} \times 2$ + AvgPool	40 M (24M+13M+3 M)

The c_{in}, c_{out}'s for the final class layer are $512 \rightarrow 1024 \rightarrow 2048$. The learnable parameters for each OptEq blocks are two depthwise separable convolution kernels.

Notably, OptEq significantly improves the results in comparison to the pure extractor in addition to classification layer (e.g., ResNet-50 (Extr.) and W-ResNet-50 (Extr.)). Moreover, even with much smaller model size, the performance of OptEq early goes beyond the bars given by the original ResNet-50 and W-ResNet-50. Hence, we can conclude that the superiority mainly comes from our OptEqs rather than extractors.

We also set different α's for different layers. In general, it is hard to tune the parameters manually. Hence we let the model learn α by itself. We initiate them to 0.8 and update them by BP. The result in Table 7 shows that the learned alpha can obtain a comparable performance with the fixed one. Although intuitively learnable α's are more reasonable for model design, it increases the difficulty of model training. Hence, we may not always get better results in this case. How to overcome the unstable behaviors of univariate variables during DNN training is still an open problem. Note that, the learnable α's converge near [0.63,0.72,0.86,0.9,0.87] finally.

7.4 Efficiency and Approximation Error

We train deep OptEqs by the IFT based way given in [3] and compare the results with the unrolling way. The time for inference and BP is provided, and it is the total time for 80 iteration steps with the batch size being 125 on GPU NVIDIA GTX 1070. The relative residual is averaged over all *test* batches: $\|\mathbf{z}^K - \mathcal{T}(\mathbf{z}^K, \mathbf{x}, \boldsymbol{\theta})\|_2 / \|\mathbf{z}^K\|_2$. For fair comparison, we do not utilize any feature regularization in this experiment. We set $\alpha = 0.8$ and let "thd" represent the residual threshold. In order to accelerate convergence speed of the iteration and stabilize OptEq, it is crucial to make sure that the operator norms of the initial weight matrices are small. Therefore, we use initialization schemes with small values (around 0). Moreover, experimental results indicate that initialization schemes with small values do not affect the performance of OptEqs.

TABLE 7
Evaluation Results on ImageNet Classification With Top-1 and Top-5 Accuracies

	Models	top1 Acc.	top5 Acc.	Size
Explicit	ResNet-18	70.2%	89.9%	13 M
	ResNet-34	74.8%	91.1%	21 M
	Inception-V2	74.8%	92.2%	12 M
	ResNet-50	75.1%	92.5%	26 M
	Res-Extr. + Cls.	71.2%	89.5%	11 M
	HRNet-W18	76.8%	**93.4%**	21 M
Implicit	MDEQ-single branch	72.9%	91.0%	18 M
	MDEQ-small	75.5%	92.7%	18 M
	OptEq (learnable α's)	76.7%	93.1%	18 M
	OptEq-small	**76.9%**	93.1%	18 M
Explicit	ResNet-101	77.1%	93.5%	52 M
	W-ResNet-50	78.1%	93.9%	69 M
	W-Res-Extr. + Cls.	73.4%	90.9%	27 M
Implicit	MDEQ-large	77.5%	93.6%	63 M
	MDEQ (Unrolled)	75.9%	93.0%	63 M
	OptEq-Mid.	**78.3%**	**94.0%**	40 M

The results are given in Table 8, although IFT based methods consume much less memory, given the comparable relative residual, the unrolling methods achieve better performance with much less inference and BP time. Note that a loose residual threshold may destroy the IFT based method significantly. We should choose the appropriate training method according to practice. For IFT, the inference time is longer than the BP one since the fixed point equation needed to solve during inference is non-linear, which is more challenging than the linear one during BP.

A more practice comparison of the efficiency on the large vision dataset is given in Table 9. We only consider the methods that release the codes on this dataset. We align all the hyper-parameters that may affect the FPS, such as fixed-point algorithms (Broyden methods), batch size (768), number of GPUs (8) and workers (8), etc. This experiment is performed in the distributed data-parallel model on 8 Tesla A100.

In general, at the cost of model size, the training speed for explicit models is much faster than the implicit models. When only considering the equilibrium methods, we can find that unrolling-based ways are much more efficient. The main reason is that the fixed point is hard to obtain when the forward function of the equilibrium model is complicated. On the other hand, without proper temporary results of computation graph, we may pay a lot of computing load to carry out BP. However, the IFT-based way is a good choice if the memory of GPUs is limited. By the way, in the case of utilizing depthwise separable convolution, our method is inefficient. If use raw convolution, our efficiency is comparable to MDEQ (we adjust some hidden layers to align the size of model). Therefore, we choose the unrolling way to train the large Opteqs.

7.5 Cityscapes Semantic Segmentation

In this experiment, we evaluate the empirical performance of our deep OptEq on a large-scale computer vision task: semantic segmentation on the Cityscapes dataset. We construct a deep OptEq with only three weighted layers and channels of 256, 512 and 512. The deep OptEq is used as the "backbone"

TABLE 8
Comparison Between Unrolling and IFT Based Training (# Params 199 k)

Method	Acc.	Inference Time	Back-Prop Time	Relative Residual
Unrolling (K=5)	83.52%	**1.3**s	**1.8**s	1.22e-02
Unrolling (K=10)	87.28%	2.2 s	3.3 s	4.88e-03
Unrolling (K=20)	87.71%	3.9 s	6.4 s	4.81e-04
Unrolling (K=40)	**87.83**%	7.5 s	12.7 s	**1.20e-05**
IFT (thd = 1e-03)	87.63%	16.3 s	6.7 s	7.33e-04
IFT (thd = 1e-02)	85.95%	15.6 s	1.3 s	9.52e-03

of the segmentation network. We compare our method with FCN [45] on the Cityscapes test set. We employ the poly learning rate policy to adjust the learning rate, where the initial learning rate is multiplied by $(1 - iter/total\_iter)^{0.9}$ after each iteration. The initial learning rate is set to be 0.01 for both networks. Moreover, momentum and weight decay are set to 0.9 and 0.001, respectively. Note that we only train on finely annotated data. We train the model for 40 K iterations, with mini-batch size set as 8. The results on the validation set are shown in Table 10. Notably, our deep OptEq significantly outperforms FCN with a similar number of parameters. Note that in this experiment, we have not introduced any customized property of the feature, so the performance improvement is entirely due to the superiority of the implicit structure of deep OptEq.

8 Conclusion

In this paper, we decompose the feed-forward DNN and find a more reasonable basic unit layer, which shows a close relationship with the proximal operator. Based on it, we propose new equilibrium models, OptEqs, and explore their underlying optimization problems thoroughly. We provide two strategies to introduce customized regularizations to the equilibrium points, and achieve significant performance improvement in experiments. We highlight that by modifying the underlying optimization problems, we can create more effective network architectures. Our work may inspire more interpretable equilibrium models from the optimization perspective.

TABLE 9
Comparison Between FPS on ImageNet

	Models	FPS (images/s)	Size
Explicit	ResNet-50	4597/s	26 M
Implicit	MDEQ-single branch	257/s	18 M
	MDEQ	1057/s	18 M
	OptEq (unrolling K=5)	1835/s	18 M
	OptEq (IFT with dw-conv)	200/s	18 M
	OptEq (IFT w/o dw-conv)	1137/s	18 M

We align all training-related hyper parameters, e.g., batch size, number of GPUs, etc. For MDEQ we set downsample to 2.

TABLE 10
Evaluation on the Validation Set of Cityscapes Semantic Segmentation

Method	mIoU	mAcc	aAcc
FCN	71.47	79.23	95.56
Deep OptEq	74.47	81.91	95.93

Acknowledgments

We also appreciate Huan Li (NKU), Yifei Wang (PKU) and Zhenyang Geng (CMU) for their valuable and detailed discussions. Xingyu Xie and Qiuhao Wang are equal contribution to this work.

References

[1] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 6572–6583.
[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[3] S. Bai, J. Z. Kolter, and V. Koltun, "Deep equilibrium models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 690–701.
[4] S. Bai, V. Koltun, and J. Z. Kolter, "Multiscale deep equilibrium models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, Art. no. 440.
[5] J. Li, M. Xiao, C. Fang, Y. Dai, C. Xu, and Z. Lin, "Training neural networks by lifted proximal operator machines," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 6, pp. 3334–3348, Jun. 2022.
[6] H. Ramsauer *et al.*, "Hopfield networks is all you need," 2020, *arXiv:2008.02217*.
[7] J. Li, C. Fang, and Z. Lin, "Lifted proximal operator machines," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, pp. 4181–4188.
[8] E. Winston and J. Z. Kolter, "Monotone operator equilibrium networks," 2020, *arXiv:2006.08591*.
[9] M. Revay, R. Wang, and I. R. Manchester, "Lipschitz bounded equilibrium networks," 2020, *arXiv:2010.01732*.
[10] X. Hong-Kun, "Viscosity approximation methods for nonexpansive mappings," *J. Math. Anal. Appl.*, vol. 298, no. 1, pp. 279–291, 2004.
[11] S. Sabach and S. Shtern, "A first order method for solving convex bilevel optimization problems," *SIAM J. Optim.*, vol. 27, no. 2, pp. 640–660, 2017.
[12] S. W. Fung, H. Heaton, Q. Li, D. McKenzie, S. Osher, and W. Yin, "Fixed point networks: Implicit depth models with jacobian-free backprop," 2021, *arXiv:2103.12803*.
[13] S. Gurumurthy, S. Bai, Z. Manchester, and J. Z. Kolter, "Joint inference and input optimization in equilibrium networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 16818–16832.
[14] K. Kawaguchi, "On the theory of implicit deep learning: Global convergence with implicit layers," 2021, *arXiv:2102.07346*.
[15] C. B. Do, C.-S. Foo, and A. Y. Ng, "Efficient multiple hyperparameter learning for log-linear models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 377–384.
[16] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil, "Bilevel programming for hyperparameter optimization and meta-learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1568–1577.
[17] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," 2018, *arXiv:1806.09055*.
[18] Z. Ramzi, F. Mannel, S. Bai, J.-L. Starck, P. Ciuciu, and T. Moreau, "SHINE: Sharing the inverse estimate from the forward pass for bilevel optimization and implicit models," 2021, *arXiv:2106.00553*.
[19] J. Lorraine, P. Vicol, and D. Duvenaud, "Optimizing millions of hyperparameters by implicit differentiation," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 1540–1552.
[20] R. Liu, J. Gao, J. Zhang, D. Meng, and Z. Lin, "Investigating bilevel optimization for learning and vision from a unified perspective: A survey and beyond," 2021, *arXiv:2101.11517*.

[21] J. Liu, X. Chen, Z. Wang, and W. Yin, "ALISTA: Analytic weights are as good as learned weights in LISTA," in *Proc. Int. Conf. Learn. Representations*, 2019.

[22] X. Xie, J. Wu, G. Liu, Z. Zhong, and Z. Lin, "Differentiable linearized ADMM," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6902–6911.

[23] K. Wei, A. Aviles-Rivero, J. Liang, Y. Fu, C.-B. Schönlieb, and H. Huang, "Tuning-free plug-and-play proximal algorithm for inverse imaging problems," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 10158–10169.

[24] B. Amos and J. Z. Kolter, "OptNet: Differentiable optimization as a layer in neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 136–145.

[25] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 9562–9574.

[26] R. A. Amjad and B. C. Geiger, "Learning representations for neural network-based classification using the information bottleneck principle," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 9, pp. 2225–2239, Sep. 2020.

[27] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," 2016, *arXiv:1612.00410*.

[28] X. Zhang, P. Cui, R. Xu, L. Zhou, Y. He, and Z. Shen, "Deep stable learning for out-of-distribution generalization," 2021, *arXiv:2104.07876*.

[29] D. Soudry, E. Hoffer, M. S. Nacson, S. Gunasekar, and N. Srebro, "The implicit bias of gradient descent on separable data," *J. Mach. Learn. Res.*, vol. 19, no. 1, pp. 2822–2878, 2018.

[30] N. Razin and N. Cohen, "Implicit regularization in deep learning may not be explainable by norms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, Art. no. 1778.

[31] A. Rajeswaran, C. Finn, S. Kakade, and S. Levine, "Meta-learning with implicit gradients," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, Art. no. 11.

[32] M. Frigon, "Fixed point and continuation results for contractions in metric and gauge spaces," *Banach Center Pub.*, vol. 77, 2007, Art. no. 89.

[33] R. Vershynin, *High-Dimensional Probability: An Introduction With Applications in Data Science*, vol. 47. Cambridge, U.K.: Cambridge Univ. Press, 2018.

[34] S. Bai, V. Koltun, and J. Z. Kolter, "Stabilizing equilibrium models by jacobian regularization," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 554–565.

[35] Z. Geng, X.-Y. Zhang, S. Bai, Y. Wang, and Z. Lin, "On training implicit models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 24247–24260.

[36] M. Takada, T. Suzuki, and H. Fujisawa, "Independently interpretable lasso: A new regularizer for sparse regression with uncorrelated variables," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2018, pp. 454–463.

[37] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[38] L. Song, A. Smola, A. Gretton, J. Bedo, and K. Borgwardt, "Feature selection via dependence maximization," *J. Mach. Learn. Res.*, vol. 13, no. 5, pp. 1393–1434, 2012.

[39] Y. Yuan, K. Yang, and C. Zhang, "Feature incay for representation regularization," 2017, *arXiv:1705.10284*.

[40] B. O. Ayinde, T. Inanc, and J. M. Zurada, "Regularizing deep neural networks by enhancing diversity in feature extraction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2650–2661, Sep. 2019.

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

[42] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[43] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*.

[44] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1251–1258.

[45] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, Apr. 2017.

[46] R. Gribonval and M. Nikolova, "A characterization of proximity operators," *J. Math. Imag. Vis.*, vol. 62, pp. 773–789, 2020.

[47] A. Beck, *First-Order Methods in Optimization*. Philadelphia, PA, USA: SIAM, 2017.

[48] H.-K. Xu, "Iterative algorithms for nonlinear operators," *J. London Math. Soc.*, vol. 66, no. 1, pp. 240–256, 2002.

[49] H. H. Bauschke *et al.*, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, vol. 408. Berlin, Germany: Springer, 2011.

Xingyu Xie (Student Member, IEEE) is currently working toward the PhD degree with the School of Artificial Intelligence, Peking University. His research interests include machine learning and optimization.

Qiuhao Wang received the BS degree from the Department of Mathematical Sciences, Tsinghua University, in 2019. He is currently working toward the master's degree with the School of Artificial Intelligence, Peking University. His research interests include deep learning and optimization.

Zenan Ling received the BS degree from the Department of Mathematics, Nanjing University, in 2015, and the PhD degree from the Department of Electrical Engineering, Shanghai Jiaotong University, in 2020. He is a postdoctoral researcher with the Key Lab. of Machine Perception, Artificial Intelligence, Peking University. He is also a postdoctoral researcher with the Pazhou Lab. His research interests include random matrix theory and machine learning.

Xia Li received the BS degree from the School of Computer Science, Beijing University of Posts and Telecommunications, in 2017, and the master's degree from the School of Electronic and Computer Engineering, Peking University, in 2020. He is currently woeking toward the PhD degree with the Department of Computer Science, ETH Zurich. His research interests include image segmentation and image translation.

Guangcan Liu (Senior Member, IEEE) received the bachelor's degree in mathematics and the PhD degree in computer science and engineering from Shanghai Jiao Tong University, Shanghai, China, in 2004 and 2010, respectively. He was a post-doctoral researcher with the National University of Singapore, Singapore, from 2011 to 2012; the University of Illinois at Urbana–Champaign, Champaign, Illinois, from 2012 to 2013; Cornell University, Ithaca, New York, from 2013 to 2014; and Rutgers University, Piscataway, New Jersey, in 2014. He was a professor with the School of Automation, Nanjing University of Information Science and Technology, Nanjing, from 2014 to 2021. He is currently a professor with the School of Automation, Southeast University, Nanjing, China. His research interests include the areas of machine learning, computer vision, and signal processing.

Zhouchen Lin (Fellow, IEEE) received the PhD degree in applied mathematics from Peking University, in 2000. He is currently a professor with the Key Laboratory of Machine Perception (MOE), School of Artificial Intelligence, Peking University. His research interests include machine learning and numerical optimization. He was area chairs of ACML, ACCV, CVPR, ICCV, NIPS/NeurIPS, AAAI, IJCAI, ICLR, and ICML many times, a program co-chair of ICPR 2022, and senior area chairs of ICML 2022 and NeurIPS 2022. He was an associate editor of the *IEEE Transactions on Pattern Analysis and Machine Intelligence* and currently is an associate editor of the *International Journal of Computer Vision*. He is fellows of the IAPR, and the CSIG.