On Training Implicit Models

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

This paper focuses on the principles of training implicit models of infinite layers. Specifically, previous works employ the implicit differentiation and solve the *exact* gradient for the backward propagation. However, *is it necessary to compute such an exact gradient (which is usually quite expensive) for training?* To this end, we propose a novel gradient estimate for these implicit models, named *phantom gradient*, that 1) forgoes the costly approximation of the exact gradient; and 2) provides an update direction (empirically) preferable to the implicit model training. We theoretically analyze the condition under which a descent direction of the loss landscape could be found, and provide two specific instantiations of the phantom gradient based on unrolling and the Neumann series. Experiments on large-scale vision tasks demonstrate that these lightweight phantom gradients significantly accelerate the backward passes in training implicit models (roughly 1.7× speedup), and even boost the performance over approaches based on the exact gradient.

1 Introduction

Conventional neural networks are typically constructed by explicitly stacking multiple linear and non-linear operators in a feed-forward manner. Recently, the implicitly-defined models [1, 2, 3, 4, 5] have attracted increasing attentions and are able to match the state-of-the-art level results by explicit models on several vision [4, 5] and natural language processing [3] tasks. Specifically, these works treat the evolution of the intermediate hidden states as a certain form of dynamical system, such as fixed-point equations [3, 4, 5] or an ordinary differential equation (ODE) flow [1, 2], which represents infinite latent states. The forward passes of implicit models are therefore formulated as solving these underlying dynamics, by either black-box ODE solvers [1, 2] or root-finding algorithms [3, 4]. As for the backward passes, however, directly differentiating through the forward pass trajectories could induce a heavy memory overhead [6, 7]. To this end, researchers have developed several approaches based on the implicit differentiation, such as solving a Jacobian-based linear fixed-point equation for the backward pass of deep equilibrium (DEQ) models [3], which eventually makes the backpropagation trajectories independent of the forward pass ones, allowing one to train these implicit models with essentially constant memory consumption (as we only need to store the final output and the layer itself, without any intermediate states).

However, in order to estimate the exact gradient promised by the implicit differentiation, these implicit models still have to rely on black-box solvers (*e.g.,* ODE solvers or root-solving algorithms), whose iterative nature usually makes the gradient computation very costly in practice (*e.g.,* over 30 steps for large-scale DEQ models). In this work, we investigate the question of whether an accurate gradient estimate is necessary for training implicit models. We found that *a first-order oracle that produces good "gradient estimates" is enough to efficiently and effectively train the model*, without the need to precisely (and laboriously) compute the exact gradient, as in prior works [3, 4, 8, 9].

As an application of our principle, we develop a framework in which a balanced trade-off is made between the precision and conditioning of the gradient estimate. Specifically, we name our gradient

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

estimate as the *phantom gradient*, and provide the general condition under which the phantom gradient can provide a *descent direction* of the loss landscape. We further propose two instantiations of the phantom gradient in the context of DEQ models, which are based on the the simple fixed-point unrolling and the Neumann series analysis. Importantly, we show that our proposed instantiations satisfy the descent condition, and the stochastic gradient descent (SGD) algorithm based on the phantom gradient enjoys a sound convergence property as long as the relevant hyperparameters (*e.g.,* the damping factor) are wisely selected. Note that our proposed method only directly affects, and thus accelerates, the backward formulation of these implicit models, with the forward pass formulation (*i.e.,* the root-solving process) and inference-time behavior mostly intact.

We conduct an extensive set of synthetic, ablation, and large-scale experiments to both analyze the theoretical properties of the phantom gradient and validate its performance on large-scale tasks, such as CIFAR-10 [10] and 224 × 224 ImageNet [11] classification tasks. Overall, our results suggest that: 1) the phantom gradient estimates a descent direction; 2) it is applicable to large-scale tasks and is capable of achieving a strong performance that is comparable with or better than when using exact gradients; and 3) it significantly shortens the training time needed for implicit models, by a factor of $1.4 \sim 1.7\times$. We believe these theoretical and empirical results provide strong evidence for the effectiveness of training implicit models with the inexact and lightweight phantom gradient.

2 Method

2.1 Inspection of Implicit Differentiation

In this work, we primarily focus on the formulation of root-solving-based implicit models, represented by the DEQ models [3]. Specifically, given a non-linear layer $\mathcal{F}$, the output of the implicit model is characterized as the solution $\boldsymbol{h}^*$ to the following fixed-point equation:

$$\boldsymbol{h}^* = \mathcal{F}(\boldsymbol{h}^*, \boldsymbol{z}), \tag{1}$$

where $\boldsymbol{z} \in \mathbb{R}^{d_{\boldsymbol{u}}+d_{\boldsymbol{\theta}}}$ is the union of the module's input $\boldsymbol{u} \in \mathbb{R}^{d_{\boldsymbol{u}}}$ and parameters $\boldsymbol{\theta} \in \mathbb{R}^{d_{\boldsymbol{\theta}}}$, *i.e.,* $\boldsymbol{z}^\top = [\boldsymbol{u}^\top, \boldsymbol{\theta}^\top]$. Here, $\boldsymbol{u}$ is usually the projection of the original data point $\boldsymbol{x} \in \mathbb{R}^{d_{\boldsymbol{x}}}$, *e.g.,* $\boldsymbol{u} = \mathcal{M}(\boldsymbol{x})$. In this section, we assume $\mathcal{F}$ is a contractive mapping *w.r.t.* $\boldsymbol{h}$ so that its Lipschitz constant $L_{\boldsymbol{h}}$ *w.r.t.* $\boldsymbol{h}$ is less than one (*i.e.,* $L_{\boldsymbol{h}} < 1$), a setting that has been analyzed in numerous prior works [12, 13, 14].

To perform backpropagation through the module induced by Eq. (1), we need to calculate the Jacobian matrix of $\boldsymbol{h}^*$ *w.r.t.* the projected input (as well as parameters) $\boldsymbol{z}$. By Implicit Function Theorem,

$$\frac{\partial \boldsymbol{h}^*}{\partial \boldsymbol{z}} = \left.\frac{\partial \mathcal{F}}{\partial \boldsymbol{z}}\right|_{\boldsymbol{h}^*} \left(\boldsymbol{I} - \left.\frac{\partial \mathcal{F}}{\partial \boldsymbol{h}}\right|_{\boldsymbol{h}^*}\right)^{-1} \tag{2}$$

The fixed point $\boldsymbol{h}^*$ of Eq. (1) is then passed to a post-processing function $\mathcal{G}$ to predict $\widetilde{\boldsymbol{y}} = \mathcal{G}(\boldsymbol{h}^*)$. In the generic learning scenario, the training objective is to minimize the following expected loss:

$$\mathcal{R}(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{P}}\left[\mathcal{L}(\widetilde{\boldsymbol{y}}(\boldsymbol{x}, \boldsymbol{\theta}), \boldsymbol{y})\right], \tag{3}$$

where $\boldsymbol{y}$ is the groundtruth corresponding to the training example $\boldsymbol{x}$, and $\mathcal{P}$ is the data distribution. Here, we omit the parameters of $\mathcal{G}$, because given the output $\boldsymbol{h}^*$ of the implicit module $\mathcal{F}$, training the post-processing part $\mathcal{G}$ is the same as training explicit neural networks. The most crucial component is the gradient of the loss function $\mathcal{L}$ *w.r.t.* the input vector $\boldsymbol{z}^\top = [\boldsymbol{u}^\top, \boldsymbol{\theta}^\top]$, which is used to train both the implicit module $\mathcal{F}$ and the input projection module $\mathcal{M}$. Using Eq. (2) with the condition $\boldsymbol{h} = \boldsymbol{h}^*$, we have

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}} = \frac{\partial \mathcal{F}}{\partial \boldsymbol{u}} \left(\boldsymbol{I} - \frac{\partial \mathcal{F}}{\partial \boldsymbol{h}}\right)^{-1} \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}}, \quad \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathcal{F}}{\partial \boldsymbol{\theta}} \left(\boldsymbol{I} - \frac{\partial \mathcal{F}}{\partial \boldsymbol{h}}\right)^{-1} \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}}. \tag{4}$$

The gradients in Eq. (4) are symmetric *w.r.t.* $\boldsymbol{u}$ and $\boldsymbol{\theta}$. Without loss of generality, we only discuss the gradient *w.r.t.* $\boldsymbol{\theta}$ in the following sections.

The most intriguing part lies in the Jacobian-inverse term, *i.e.,* $(\boldsymbol{I} - \partial\mathcal{F}/\partial\boldsymbol{h})^{-1}$. Because of the inverse operation, a natural question arises from the numerical aspect. *Is it well-conditioned?* If the absolute value of the eigenvalue of $\partial\mathcal{F}/\partial\boldsymbol{h}$ is close to 1, the Jacobian-inverse will be numerically unstable. This spurs us to rethink the necessity of the Jacobian-inverse term in the standard implicit differentiation. Note that the Jacobian-inverse is calculated to update model parameters with their gradients, but the exact gradient is not always optimal in model training. For example, previous research has instead used a moderate gradient noise as a regularization approach [15], which have been shown to play a central role in escaping poor local minima and improving generalization ability [16, 17, 18]. Moreover, as computing the inverse of a large matrix (e.g., $\partial \mathcal{F}/\partial \boldsymbol{h}$ is more than $10^5 \times 10^5$ on ImageNet) is intractable, prior works [3] proposes to iteratively solve a linear system involving a Jacobian-vector product instead, which makes the actual backward pass slow.

These observations motivate us to design an inexact, but theoretically sound and practically efficient gradient for training implicit models. Here, suppose the Jacobian $\partial \boldsymbol{h}^*/\partial \boldsymbol{\theta}$ is replaced with a matrix $\boldsymbol{A}$, and the corresponding *phantom gradient* is defined as

$$\widehat{\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}} := \boldsymbol{A}\,\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}}. \tag{5}$$

Next, we give the general condition on $\boldsymbol{A}$ so that a descent property of the phantom gradient can be guaranteed (Sec. 2.2), and provide two concrete instantiations of $\boldsymbol{A}$ based on either fixed-point unrolling or the Neumann series (Sec. 2.3).

2.2 General Condition on the Phantom Gradient

The following theorem formulates a sufficient condition that the phantom gradient gives a descent direction of the loss landscape. Please refer to the appendix for the proof.

Theorem 1. *Suppose the exact gradient and the phantom gradient are given by Eq.* (4) *and Eq.* (5)*, respectively. Let* σ_{max} *and* σ_{min} *be the maximal and minimal singular value of* $\partial \mathcal{F}/\partial \boldsymbol{\theta}$*. If*

$$\left\| \boldsymbol{A}\left(\boldsymbol{I} - \frac{\partial \mathcal{F}}{\partial \boldsymbol{h}}\right) - \frac{\partial \mathcal{F}}{\partial \boldsymbol{\theta}} \right\| \leq \frac{\sigma_{min}^2}{\sigma_{max}}, \tag{6}$$

then the phantom gradient provides a descent direction of the function $\mathcal{L}$*, i.e.,*

$$\left\langle \widehat{\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}}, \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} \right\rangle \geq 0. \tag{7}$$

Remark 1. Suppose only the $\left(\boldsymbol{I} - \partial \mathcal{F}/\partial \boldsymbol{h}\right)^{-1}$ term is replaced with a matrix $\boldsymbol{D}$, namely, $\boldsymbol{A} = \left(\partial \mathcal{F}/\partial \boldsymbol{\theta}\right) \boldsymbol{D}$. Then, the condition in (6) can be reduced into

$$\left\| \boldsymbol{D}\left(\boldsymbol{I} - \frac{\partial \mathcal{F}}{\partial \boldsymbol{h}}\right) - \boldsymbol{I} \right\| \leq \frac{1}{\kappa^2}, \tag{8}$$

where κ is the condition number of $\partial \mathcal{F}/\partial \boldsymbol{\theta}$. The derivation can be found in the appendix.

Remark 2. The singular value σ and the condition number κ of $\partial \mathcal{F}/\partial \boldsymbol{\theta}$ make it tricky to ensure the condition in (6) or (8). However, with $\boldsymbol{J} = \partial \mathcal{F}/\partial \boldsymbol{\theta}$, $\boldsymbol{\Theta} = \boldsymbol{J}^\top \boldsymbol{J}$ is exactly the *neural tangent kernel* (NTK) [19] corresponding to the module $\mathcal{F}$. If $\mathcal{F}$ is a multi-layer neural network, its NTK $\boldsymbol{\Theta}$ converges in probability to a scalar matrix in the infinite-width limit, *i.e.,*

$$\boldsymbol{\Theta} \xrightarrow{P} s\boldsymbol{I}, \quad \text{for some } s \in \mathbb{R}_+, \text{ as width} \to \infty. \tag{9}$$

This conclusion holds both at initialization and during the training process [19, Theorem 1 & 2][1], which implies that if $\mathcal{F}$ is sufficiently wide, all singular values of $\boldsymbol{J}$ and thus its condition number κ are close to 1 in the entire training stage. This property makes the threshold in (6) and (8) computable.

2.3 Instantiations of the Phantom Gradient

In this section, we present two practical instantiations of the phantom gradient. We also verify that the general condition in Theorem 1 can be satisfied if the hyperparameters in our instantiations are wisely selected.

---

[1] A concrete bound of the approximation error for finite-width networks can be found in [20, Theorem 3.1 & 3.2].

Suppose we hope to differentiate through an implicit dynamic, *e.g.*, either a root-solving process or an optimization problem. Previous solutions towards this include differentiating through the unrolled steps of the dynamics [21] and using the Neumann series [7]. In our case, if we solve the root of Eq. (1) via fixed-point iteration:

$$\boldsymbol{h}_{t+1} = \mathcal{F}(\boldsymbol{h}_t, \boldsymbol{z}), \quad t = 0, 1, \cdots, T-1, \tag{10}$$

then by differentiating through the unrolled steps of Eq. (10), we have

$$\frac{\partial \boldsymbol{h}_T}{\partial \boldsymbol{\theta}} = \sum_{t=0}^{T-1} \left.\frac{\partial \mathcal{F}}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{h}_t} \prod_{s=t+1}^{T-1} \left.\frac{\partial \mathcal{F}}{\partial \boldsymbol{h}}\right|_{\boldsymbol{h}_s}. \tag{11}$$

Besides, the Neumann series of the Jacobian-inverse $\left(\boldsymbol{I} - \partial \mathcal{F}/\partial \boldsymbol{h}\right)^{-1}$ is

$$\boldsymbol{I} + \frac{\partial \mathcal{F}}{\partial \boldsymbol{h}} + \left(\frac{\partial \mathcal{F}}{\partial \boldsymbol{h}}\right)^2 + \left(\frac{\partial \mathcal{F}}{\partial \boldsymbol{h}}\right)^3 + \cdots. \tag{12}$$

Notably, computing the Jacobian $\partial \boldsymbol{h}^*/\partial \boldsymbol{\theta}$ using the Neumann series in (12) is equivalent to differentiating through the unrolled steps of Eq. (10) at the exact solution point $\boldsymbol{h}^*$ and taking the limit of infinite steps [7].

Without altering the root of Eq. (1), we consider a damped variant of the fixed-point iteration:

$$\boldsymbol{h}_{t+1} = \lambda \mathcal{F}(\boldsymbol{h}_t, \boldsymbol{z}) + (1-\lambda)\boldsymbol{h}_t, \quad t = 0, 1, \cdots, T-1. \tag{13}$$

Differentiating through the unrolled steps of Eq. (13), Eq. (11) is adapted as

$$\frac{\partial \boldsymbol{h}_T}{\partial \boldsymbol{\theta}} = \lambda \sum_{t=0}^{T-1} \left.\frac{\partial \mathcal{F}}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{h}_t} \prod_{s=t+1}^{T-1} \left(\lambda \left.\frac{\partial \mathcal{F}}{\partial \boldsymbol{h}}\right|_{\boldsymbol{h}_s} + (1-\lambda)\, \boldsymbol{I} \right). \tag{14}$$

The Neumann series of $\left(\boldsymbol{I} - \partial \mathcal{F}/\partial \boldsymbol{h}\right)^{-1}$ is correspondingly adapted as

$$\lambda \left(\boldsymbol{I} + \boldsymbol{B} + \boldsymbol{B}^2 + \boldsymbol{B}^3 + \cdots\right), \quad \text{where } \boldsymbol{B} = \lambda \frac{\partial \mathcal{F}}{\partial \boldsymbol{h}} + (1-\lambda)\boldsymbol{I}. \tag{15}$$

The next theorem shows that under mild conditions, the Jacobian in Eq. (14) converges to the exact Jacobian and the Neumann series in (15) converges to the Jacobian-inverse $\left(\boldsymbol{I} - \partial \mathcal{F}/\partial \boldsymbol{h}\right)^{-1}$.

Theorem 2. *Suppose the matrix $\partial \mathcal{F}/\partial \boldsymbol{h}$ is a contractive mapping. Then,*

(i) the Neumann series in (15) converges to the Jacobian-inverse $\left(\boldsymbol{I} - \partial \mathcal{F}/\partial \boldsymbol{h}\right)^{-1}$; and

(ii) if the function $\mathcal{F}$ is continuously differentiable w.r.t. both $\boldsymbol{h}$ and $\boldsymbol{\theta}$, the sequence in Eq. (14) converges to the exact Jacobian $\partial \boldsymbol{h}^*/\partial \boldsymbol{\theta}$ as $T \to \infty$, i.e.,

$$\lim_{T \to \infty} \frac{\partial \boldsymbol{h}_T}{\partial \boldsymbol{\theta}} = \left.\frac{\partial \mathcal{F}}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{h}^*} \left(\boldsymbol{I} - \left.\frac{\partial \mathcal{F}}{\partial \boldsymbol{h}}\right|_{\boldsymbol{h}^*} \right)^{-1}. \tag{16}$$

However, as discussed in Sec. 2.1, it is unnecessary to compute the exact gradient with infinite terms. In the following context, we introduced two instantiations of the phantom gradient based on a finite-term truncation of Eq. (14) or (15).

Unrolling-based Phantom Gradient. In the unrolling form, the matrix $\boldsymbol{A}$ is defined as

$$\boldsymbol{A}^{\text{unr}}_{k,\lambda} = \lambda \sum_{t=0}^{k-1} \left.\frac{\partial \mathcal{F}}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{h}_t} \prod_{s=t+1}^{k-1} \left(\lambda \left.\frac{\partial \mathcal{F}}{\partial \boldsymbol{h}}\right|_{\boldsymbol{h}_s} + (1-\lambda)\, \boldsymbol{I} \right). \tag{17}$$

Neumann-series-based Phantom Gradient. In the Neumann form, the matrix $\boldsymbol{A}$ is defined as

$$\boldsymbol{A}^{\text{neu}}_{k,\lambda} = \lambda \left.\frac{\partial \mathcal{F}}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{h}^*} \left(\boldsymbol{I} + \boldsymbol{B} + \boldsymbol{B}^2 + \cdots + \boldsymbol{B}^{k-1}\right), \quad \text{where } \boldsymbol{B} = \lambda \left.\frac{\partial \mathcal{F}}{\partial \boldsymbol{h}}\right|_{\boldsymbol{h}^*} + (1-\lambda)\boldsymbol{I}. \tag{18}$$

According to Theorem 2, the matrix $\boldsymbol{A}$ defined by either Eq. (17) or Eq. (18) converges to the exact Jacobian $\partial \boldsymbol{h}^*/\partial \boldsymbol{\theta}$ as $k \to \infty$ for any $\lambda \in (0, 1]$. Therefore, by Theorem 2, the condition in (6) can be satisfied if a sufficiently large step k is selected, since

$$\left\| \boldsymbol{A}\left(\boldsymbol{I} - \frac{\partial \mathcal{F}}{\partial \boldsymbol{h}}\right) - \frac{\partial \mathcal{F}}{\partial \boldsymbol{\theta}} \right\| \le (1 + L_{\boldsymbol{h}}) \left\| \boldsymbol{A} - \frac{\partial \mathcal{F}}{\partial \boldsymbol{\theta}}\left(\boldsymbol{I} - \frac{\partial \mathcal{F}}{\partial \boldsymbol{h}}\right)^{-1} \right\|. \tag{19}$$

Next, we characterize the impact of the two hyperparameters, *i.e.,* k and λ, on the precision and conditioning of $\boldsymbol{A}$. Take the Neumann-series-based phantom gradient (Eq. (18)) as an example.

(i) On the precision of the phantom gradient,
  - a large k makes the gradient estimate more accurate, as higher order terms of the Neumann series are included; while
  - a small λ slows down the convergence of the Neumann series because of the larger norm of $\boldsymbol{B}$ with the decrease of λ.

(ii) On the conditioning of the phantom gradient,
  - a large k impairs the conditioning of $\boldsymbol{A}$ since the condition number of $\boldsymbol{B}^k$ grows exponentially as k increases; while
  - a small λ helps maintain a small condition number of $\boldsymbol{A}$ because the singular values of $\partial \mathcal{F}/\partial \boldsymbol{h}$ are "smoothed" by the identity matrix.

In a word, a large k is preferable for a more accurate $\boldsymbol{A}$, while a small λ contributes to the well-conditioning of $\boldsymbol{A}$. Practically, these hyperparameters should be selected in consideration of a balanced trade-off between the precision and conditioning of $\boldsymbol{A}$. See Sec. 3 for experimental results.

2.4 Convergence Theory

In this section, we provide convergence guarantee of the SGD algorithm using the phantom gradient. We prove that under mild conditions, if the approximation error of the phantom gradient is sufficiently small, the SGD algorithm converges to an ϵ-approximate stationary point in the expectation sense. Please refer to the appendix for the proof, where we also discuss the feasibility of our assumptions.

Theorem 3. *Suppose the loss function $\mathcal{R}$ in Eq. (3) is ℓ-smooth, lower-bounded, and has bounded gradient almost surely in the training process. Besides, assume the gradient in Eq. (4) is an unbiased estimator of $\nabla \mathcal{R}(\boldsymbol{\theta})$ with a bounded covariance. If the phantom gradient in Eq. (5) is an ϵ-approximation to the gradient in Eq. (4), i.e.,*

$$\left\| \widehat{\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} \right\| \le \epsilon, \quad \textit{almost surely}, \tag{20}$$

then using Eq. (5) as a stochastic first-order oracle with a step size of $\eta_\tau = O(1/\sqrt{\tau})$ to update $\boldsymbol{\theta}$ with gradient descent, it follows after T iterations that

$$\mathbb{E}\left[\frac{\sum_{\tau=1}^{T} \eta_\tau \|\nabla \mathcal{R}(\boldsymbol{\theta}_\tau)\|^2}{\sum_{\tau=1}^{T} \eta_\tau}\right] \le O\left(\epsilon + \frac{\log T}{\sqrt{T}}\right). \tag{21}$$

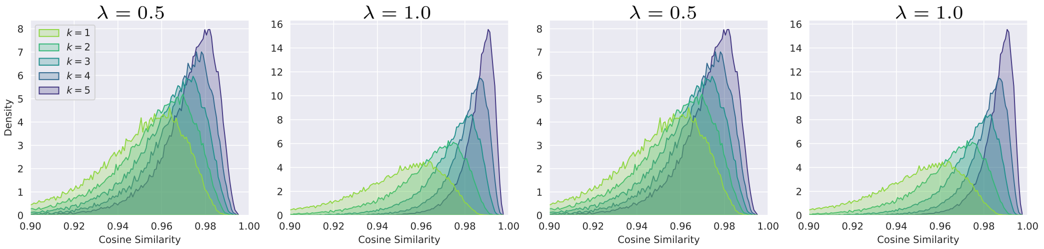
Remark 3. Consider the condition in (20):

$$\left\| \widehat{\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} \right\| \le \left\| \boldsymbol{A} - \frac{\partial \mathcal{F}}{\partial \boldsymbol{\theta}}\left(\boldsymbol{I} - \frac{\partial \mathcal{F}}{\partial \boldsymbol{h}}\right)^{-1} \right\| \left\| \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}} \right\|. \tag{22}$$

Suppose the norm of $\partial \mathcal{L}/\partial \boldsymbol{h}$ are almost-surely bounded. By Theorem 2, the condition in (20) can be guaranteed as long as a sufficiently large k is selected.

3 Experiments

In this section, we aim to answer the following questions via empirical results: (1) Does the phantom gradient form a descent direction in the practical scenario? (2) What is the difference between the



(a) Neumann-series-based phantom gradient

(b) Unrolling-based phantom gradient

Figure 1: Cosine similarity between the phantom and the exact gradients in the synthetic setting.

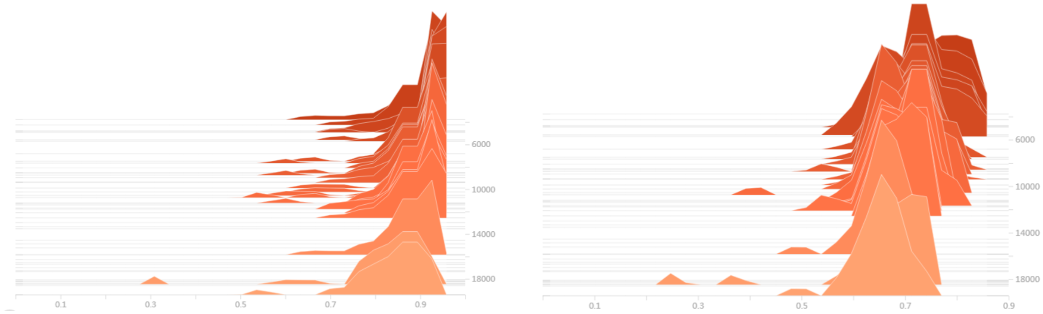
phantom gradients in the unrolling form and in the Neumann form? (3) How is the phantom gradient influenced by the hyperparameters k and λ? (4) How about the memory and computation cost of the phantom gradient compared with implicit differentiation? (5) Can the phantom gradient work at large-scale settings?

We have provided some theoretical analysis and intuitions to (1), (2), and (3) in Sec. 2.3. Now we answer (1) and (2) and demonstrate the performance curves under different hyperparameters k and λ on the CIFAR-10 dataset [10]. Besides, we also study other factors that have potential influences on the training process of the state-of-the-art implicit models [3, 4]. For (4) and (5), we conduct experiments on the large-scale datasets, including the CIFAR-10 and ImageNet [22] datasets.

We start by introducing two settings of experiments. On the CIFAR-10 dataset, we first use the MDEQ-Tiny [4] model (170K parameters) as the backbone model in an *ablation setting*. Additionally, we adopt a single-layer neural network with spectral normalization [23] as the function $\mathcal{F}$ and the fixed-point iteration as solver of $\boldsymbol{h}^*$, which is the *synthetic setting*. Besides, the unrolling-based and Neumann-series-based phantom gradients are abbreviated to UPG and NPG, respectively.

Precision of the Phantom Gradient. The precision of the phantom gradient is measured by its angle (or cosine similarity) against the exact gradient. We discuss its precision in both the synthetic setting and the ablation setting.

In the synthetic setting, the function $\mathcal{F}$ is restricted to be a contractive mapping. Specifically, we directly set the Lipschitz constant of $\mathcal{F}$ as $L_{\boldsymbol{h}} = 0.9$, and use 100 fixed-point iterations to solve the root $\boldsymbol{h}^*$ of Eq. (1) until the relative error satisfies $\|\boldsymbol{h} - \mathcal{F}(\boldsymbol{h}, \boldsymbol{z})\| / \|\boldsymbol{h}\| < 10^{-5}$. Here, the exact gradient is estimated by backpropagation through those fix point iterations, and cross-validated by implicit differentiation solved with 20 iterations of the Broyden's method [24]. In our experiment, the cosine similarity between these two gradient estimates consistently succeeds 0.9999, indicating the gradient estimate is quite accurate. The cosine similarity between the phantom gradient and the exact gradient is shown in Fig. 1. It can be seen that the cosine similarity tends to increase as k grows, and that a small λ tends to slows down the convergence of the phantom gradient, allowing it to explore in a wider range with regard to its angle against the exact gradient.



(a) Neumann-series-based phantom gradient

(b) Unrolling-based phantom gradient

Figure 2: Cosine similarity between the phantom gradient and the exact gradient in the real scenario. The x-axis corresponds to the cosine similarity, and the y-axis to the training steps. This figure characterizes the evolution of the phantom gradient's precision during the training process.

In the ablation setting, the precision of the phantom gradient during the training process is shown in Fig. 2. The model is trained by implicit differentiation under its official schedule[2]. It can be seen that in real scenarios, the phantom gradient still provides a descent direction, as indicated by the large cosine similarity against the exact gradient.

To Pretrain, or not to Pretrain? To understand the components of training implicit models via implicit differentiation, we first show a detailed ablation study of the baseline models. All performances are based on 6 independent runs in the ablation setting, and the average and best accuracy are reported in Tab. 1.

Table 1: Ablation settings on CIFAR-10.

Method	**Acc(%)**
MDEQ-Tiny + Implicit	85.0(85.3)
w/o Pretraining	82.3(84.6)
w/o Dropout	83.7(84.0)
Adam → SGD	84.4(84.8)
SGD w/o Pretrainig	81.9(85.6)
UPG ($\boldsymbol{A}_{5,0.5}$, w/o Dropout)	**85.8(86.6)**
NPG ($\boldsymbol{A}_{5,0.5}$, w/o Dropout)	85.6(86.1)
UPG ($\boldsymbol{A}_{9,0.5}$, w/ Dropout)	**86.1(87.3)**

The MDEQ model employs a pretraining stage in which the model $\mathcal{F}$ is unrolled as a recurrent network. We study the impacts of the pretraining stage, the Dropout [25] operation, and the optimizer, respectively. It can be seen that the unrolled pretraining stabilizes the training of the MDEQ model. Removing the pretraining stage leads to a large performance drop and apparent training instability among different runs because the solver cannot obtain an accurate fixed point $\boldsymbol{h}^*$ when the model is not adequately trained. This also suggests that the MDEQ model is a strong baseline for our method to compare with.

However, pretraining is not always indispensable for training implicit models. It introduces a hyperparameter that how many steps should be used in pretraining. In the later paragraph, we discuss that how the unrolling-based phantom gradient can circumvent this issue.

Trade-offs between Unrolling and Neumann. For an exact fixed point $\boldsymbol{h}^*$, *i.e.,* $\boldsymbol{h}^* = \mathcal{F}(\boldsymbol{h}^*, \boldsymbol{z})$, there is no difference between the unrolling-based phantom gradient and Neumann-series-based one. However, when the numerical error exists in solving $\boldsymbol{h}^*$, *i.e.,* $\|\boldsymbol{h}^* - \mathcal{F}(\boldsymbol{h}^*, \boldsymbol{z})\| > 0$, phantom gradients in the two forms can have different behaviors. As in Fig. 1, the unrolling-based phantom gradient demonstrates greater robustness and higher tolerance to numerical errors in the backward computation.

Table 2: Complexity comparison. **Mem** means memory cost. $K \gg k \approx 1$, where K corresponds to the solver's steps and k denotes the unrolling/Neumann steps.

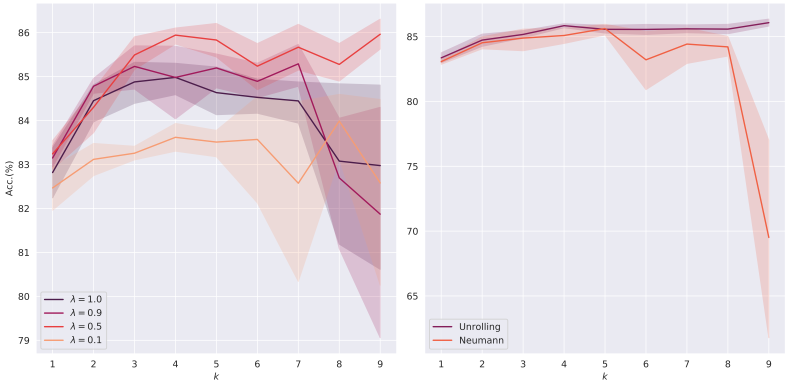
Method	**Time**	**Mem**	**Peak Mem**
Implicit	$\mathcal{O}(K)$	$\mathcal{O}(1)$	$\mathcal{O}(k)$
UPG	$\mathcal{O}(k)$	$\mathcal{O}(k)$	$\mathcal{O}(k)$
NPG	$\mathcal{O}(k)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

We note that a particular benefit of the unrolling-based phantom gradient is its ability to automatically switch between the pretraining and training stages for the MDEQ model. When the model is not sufficiently trained and the solver possibly converges poorly (see [4]), the unrolling-based phantom gradient defines a forward computational graph that is essentially equivalent to a shallow recurrent network. In this stage, the phantom gradient serves as a backpropagation through time (BPTT) algorithm and hence behaves as in the pretraining stage. Then, as training progresses, the solver becomes more stable and converges to the fixed point $\boldsymbol{h}^*$ better. This makes the unrolling-based phantom gradient behave more like the Neumann-series-based counterpart. Therefore, the unrolled pretraining is gradually transited into the regular training based on implicit differentiation, and the hyperparameter tuning of pretraining steps can be waived. We argue that such an ability to adaptively switch training stages is crucial to the implicit training protocol, which is also supported by the performance gain in Tab. 1.

Although the unrolling-based phantom gradient requires higher memory than implicit differentiation or the Neumann-series-based one, it does not surpass the peak memory usage in the entire training process of implicit differentiation due to the pretraining stage. In the ablation setting, the MDEQ [4] model employs a 10-layer unrolling for pretraining, which actually consumes double memory compared with a 5-step unrolling scheme (*e.g.,* $\boldsymbol{A}_{5,0.5}$ in Tab. 1).

---

[2] Code available at `https://github.com/locuslab/mdeq`.



(a) Impact of λ and k (b) NPG *v.s.* UPG

Figure 3: Ablation studies on (a) the hyperparameters λ and k, and (b) two forms of phantom gradient.

The performance curves in Fig. 3 further validates the advantages of the unrolling-based phantom gradient over the Neumann-series-based one and shows the influences of λ and k. In Tab. 2, we also demonstrate the time and memory complexity for implicit differentiation and the two forms of phantom gradient. In the following paragraph, we adopt the unrolling-based phantom gradient in the large-scale experiments.

Phantom Gradient at Scale. We conduct large-scale experiments to verify the advantages of the phantom gradient on CIFAR-10 and ImageNet classification benchmark. The results are illustrated in Tab. 3. Our method matches or surpasses the implicit differentiation training protocol on the state-of-the-art implicit models with a visible reduction in the training time and comparable or less peak memory usage.

Table 3: Large-scale experiments on CIFAR-10 and ImageNet classifications. Using phantom gradients, we are able to achieve comparable or better performance in these high-dimensional settings, while being much faster at training.

Task	Method	Params	Acc(%)	Speed	Peak Mem
CIFAR-10	MDEQ + Implicit	10M	93.8	1×	1×
CIFAR-10	MDEQ + UPG $\boldsymbol{A}_{5,0.5}$	10M	95.0	1.4×	0.5×
ImageNet	MDEQ + Implicit	18M	75.3	1×	1×
ImageNet	MDEQ + UPG $\boldsymbol{A}_{5,0.5}$	18M	75.1	1.7×	1×

4 Related Work

Implicit Models. Implicit models generalize the recursive forward/backward rules of neural networks and characterize their internal mechanism by some pre-specified dynamics. Based on the dynamics, the implicit models can be broadly categorized into three classes: ODE-based [1, 2], root-solving-based [3, 4, 8, 5], and optimization-based [26, 27, 28, 29] implicit models. The ODE-based implicit models [1, 2] treat the iterative update rules of residual networks as the Euler discretization of an ODE, which could be solved by any black-box ODE solver. The gradient of the differential equation is calculated using the *adjoint method* [30], in which the adjoint state is obtained by solving another ODE. The root-solving-based implicit models [3, 4, 8, 5] characterize layers of neural networks by the process of fixed-point equation solving. The equations are solved by either the black-box root-finding solver [3, 4] or the fixed-point iteration [5]. The optimization-based implicit models [26, 27, 28, 29] leverage the optimization programs as layers of neural networks. Previous work has studied differentiable layers of quadratic programming [26], submodular optimization [27], and maximum satisfiability (MAXSAT) problems [28]. As for the backward passes, implicit differentiation is applied to the problem-defining equations of the root-solving-based models [3, 4] or the KKT conditions of the optimization-based models [26]. As such, the gradient can be obtained by solving a linear system.

In this work, we focus on the root-solving-based implicit models. We differ from previous work in that we look into the theoretical aspect of the gradient-based algorithm in training implicit models. We show that besides the precision of the gradient estimate, its condition is also of great significance for the training stability. With these considerations, we show that implicit models of the same architecture could enjoy faster convergence and better generalization ability in practical applications.

Non-End-to-End Optimization in Deep Learning. Non-end-to-end optimization aims to replace the standard gradient-based training of deep architectures with modular or weakly modular training without the entire forward and backward passes. Currently, there are mainly three research directions in this field, namely, the auxiliary variable methods [31, 32, 33, 34, 35, 36, 37], target propagation [38, 39, 40], and synthetic gradient [41, 42, 43]. The auxiliary variable methods [31, 32, 33, 34, 35, 36, 37] formulate the optimization of neural networks as constrained optimization problems, in which the layer-wise activations are considered as trainable auxiliary variables. Then, the equality constraints are relaxed as penalty terms added to the objectives so that the parameters and auxiliary variables can be divided into blocks and thus optimized in parallel. The target propagation method [38, 39, 40] trains each module by having its activations regress to the pre-assigned targets, which are propagated backwards from the downstream modules. Specifically, the auto-encoder architecture is used to reconstruct targets at each layer. Finally, the synthetic gradient method [41, 42, 43] estimates the local gradient of neural networks using auxiliary models, and employ the synthetic gradient in place of the exact gradient to perform parameter update. In this way, the forward and backward passes are decoupled and can be executed in an asynchronous manner.

Our work is in line with the non-end-to-end optimization research since we also aims to decouple the forward and backward passes of neural networks. However, we show that finding a reasonable "target" or a precise gradient estimate is not always the first principle in training deep architectures. Our paper paves a path that an inexact but well-conditioned gradient estimate can contribute to both training and generalization of implicit models.

Differentiation through Implicit Dynamics. Differentiation through certain implicit dynamics is an important aspect in a wide range of research fields, including bilevel optimization [21, 7], meta-learning [44, 45, 46], and sensitivity analysis [47]. Since the gradient (or Jacobian) usually cannot be computed analytically, researchers have to implicitly differentiate the dynamics at the converged point. The formula of the gradient typically contains a term of Jacobian-inverse (or Hessian-inverse), which is computationally prohibitive for large-scale models. (See Eq. (2) in our case.) Herein, several techniques have been developed to approximate the matrix inverse in the previous literature.

An intuitive solution is to differentiate through the unrolled steps of a numerical solver of the dynamics [48, 49, 6]. In particular, if a single step is unrolled, it reduces to the well-known *one-step gradient* [50, 44, 51, 46, 52], in which the Jacobian-inverse is simply approximated by an identity matrix. On the contrary, unrolling a small number of steps may induce a bias [7], while the memory and computational cost grows linearly as the number of unrolled steps increases. Towards this issue, Shaban *et al.* [21] propose to truncate the long-term dependencies and differentiate through only the last L steps. Furthermore, if the dynamics has converged to a stationary point, the approximation in Shaban *et al.* [21] is exactly the Neumann approximation of the Jacobian-inverse with the first L terms. Based on this, Lorraine *et al.* [7] choose to directly use the truncated Neumann series as an approximation of the Jacobian-inverse. Besides the unrolling-based methods, optimization-based approaches [53, 45] have also been studied in this field. Since the Jacobian-inverse-vector product can be viewed as solution of a linear system, algorithms like the conjugate gradient method can be used to solve it.

5 Conclusion

In this work, we question the necessity of rigorously estimating the exact gradient for training implicit models. To back up our claim, we systematically analyze the general condition of a gradient estimate so that the implicit models can be guaranteed to converge to an approximate stationary point of the loss function. Specifically, we give a sufficient condition under which a first-order oracle could always find a descent direction of the loss landscape in the training process. Moreover, we introduce two instantiations of the proposed phantom gradient, based on either the fixed-point unrolling or the Neumann series. The proposed method shows $1.4 \sim 1.7\times$ accelerations with comparable or better performances on large-scale benchmarks. Overall, this paper provides an interesting and practical perspective on training implicit models with theoretical guarantees.

References

[1] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural Ordinary Differential Equations. In *Neural Information Processing Systems (NeurIPS)*, 2018. 1, 8

[2] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented Neural ODEs. In *Neural Information Processing Systems (NeurIPS)*, 2019. 1, 8

[3] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep Equilibrium Models. In *Neural Information Processing Systems (NeurIPS)*, 2019. 1, 2, 3, 6, 8

[4] Shaojie Bai, Vladlen Koltun, and J. Zico Kolter. Multiscale Deep Equilibrium Models. In *Neural Information Processing Systems (NeurIPS)*, pages 5238–5250, 2020. 1, 6, 7, 8

[5] Samy Wu Fung, Howard Heaton, Qiuwei Li, Daniel McKenzie, Stanley J. Osher, and Wotao Yin. Fixed Point Networks: Implicit Depth Models with Jacobian-Free Backprop. *arXiv preprint arXiv:2103.12803*, 2021. 1, 8

[6] Luca Franceschi, Michele Donini, Paolo Frasconi, and Massimiliano Pontil. Forward and Reverse Gradient-Based Hyperparameter Optimization. In *International Conference on Machine Learning (ICML)*, pages 1165–1173, 2017. 1, 9

[7] Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing Millions of Hyperparameters by Implicit Differentiation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1540–1552, 2020. 1, 4, 9

[8] Ezra Winston and J. Zico Kolter. Monotone operator equilibrium networks. In *Neural Information Processing Systems (NeurIPS)*, pages 10718–10728, 2020. 1, 8

[9] Cheng Lu, Jianfei Chen, Chongxuan Li, Qiuhao Wang, and Jun Zhu. Implicit normalizing flows. In *International Conference on Learning Representations (ICLR)*, 2021. 1

[10] Alex Krizhevsky, Geoffrey Hinton, et al. Learning Multiple Layers of Features from Tiny Images. Technical report, Citeseer, 2009. 2, 6

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. Ieee, 2009. 2

[12] Samy Wu Fung, Howard Heaton, Qiuwei Li, Daniel Mckenzie, S. Osher, and W. Yin. Fixed point networks: Implicit depth models with jacobian-free backprop. *ArXiv*, abs/2103.12803, 2021. 2

[13] Chirag Pabbaraju, Ezra Winston, and J. Zico Kolter. Estimating lipschitz constants of monotone deep equilibrium models. In *International Conference on Learning Representations (ICLR)*, 2021. 2

[14] Max Revay, Ruigang Wang, and Ian R Manchester. Lipschitz bounded equilibrium networks. *arXiv:2010.01732*, 2020. 2

[15] Xavier Gastaldi. Shake-Shake regularization of 3-branch residual networks. In *International Conference on Learning Representations Workshop Track*, 2017. 3

[16] Guozhong An. The Effects of Adding Noise During Backpropagation Training on a Generalization Performance. *Neural Computation*, 8(3):643–674, 1996. 3

[17] Zhanxing Zhu, Jingfeng Wu, Bing Yu, Lei Wu, and Jinwen Ma. The Anisotropic Noise in Stochastic Gradient Descent: Its Behavior of Escaping from Sharp Minima and Regularization Effects. In *International Conference on Machine Learning (ICML)*, pages 7654–7663, 2019. 3

[18] Jingfeng Wu, Wenqing Hu, Haoyi Xiong, Jun Huan, Vladimir Braverman, and Zhanxing Zhu. On the Noisy Gradient Descent that Generalizes as SGD. In *International Conference on Machine Learning (ICML)*, pages 10367–10376, 2020. 3

[19] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In *Neural Information Processing Systems (NeurIPS)*, pages 8571–8580, 2018. 3

[20] Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Russ R Salakhutdinov, and Ruosong Wang. On Exact Computation with an Infinitely Wide Neural Net. In *Neural Information Processing Systems (NeurIPS)*, pages 8139–8148, 2019. 3

[21] Amirreza Shaban, Ching-An Cheng, Nathan Hatch, and Byron Boots. Truncated Back-propagation for Bilevel Optimization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1723–1732, 2019. 4, 9

[22] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal on Computer Vision (IJCV)*, 115(3):211–252, 2015. 6

[23] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral Normalization for Generative Adversarial Networks. In *International Conference on Learning Representations (ICLR)*, 2018. 6

[24] Charles G Broyden. A Class of Methods for Solving Nonlinear Simultaneous Equations. *Mathematics of computation*, 19(92):577–593, 1965. 6

[25] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012. 7

[26] Brandon Amos and J. Zico Kolter. OptNet: Differentiable Optimization as a Layer in Neural Networks. In *International Conference on Machine Learning (ICML)*, pages 136–145, 2017. 8

[27] Josip Djolonga and Andreas Krause. Differentiable Learning of Submodular Models. *Neural Information Processing Systems (NeurIPS)*, pages 1013–1023, 2017. 8

[28] Po-Wei Wang, Priya Donti, Bryan Wilder, and Zico Kolter. SATNet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *International Conference on Machine Learning (ICML)*, pages 6545–6554, 2019. 8

[29] Marin Vlastelica, Anselm Paulus, Vít Musil, Georg Martius, and Michal Rolínek. Differentiation of Blackbox Combinatorial Solvers. In *International Conference on Learning Representations (ICLR)*, 2020. 8

[30] VG Boltyanskiy, Revaz V Gamkrelidze, YEF Mishchenko, and LS Pontryagin. Mathematical theory of optimal processes. 1962. 8

[31] Miguel Carreira-Perpinan and Weiran Wang. Distributed optimization of deeply nested systems. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 10–19, 2014. 9

[32] Gavin Taylor, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankit Patel, and Tom Goldstein. Training Neural Networks Without Gradients: A Scalable ADMM Approach. In *International Conference on Machine Learning (ICML)*, pages 2722–2731, 2016. 9

[33] Ziming Zhang, Yuting Chen, and Venkatesh Saligrama. Efficient Training of Very Deep Neural Networks for Supervised Hashing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1487–1495, 2016. 9

[34] Ziming Zhang and Matthew Brand. Convergent Block Coordinate Descent for Training Tikhonov Regularized Deep Neural Networks. In *Neural Information Processing Systems (NeurIPS)*, 2017. 9

[35] Jinshan Zeng, Shikang Ouyang, Tim Tsz-Kit Lau, Shaobo Lin, and Yuan Yao. Global convergence in deep learning with variable splitting via the Kurdyka-łojasiewicz property. *arXiv preprint arXiv:1803.00225*, 2018. 9

[36] Jia Li, Cong Fang, and Zhouchen Lin. Lifted Proximal Operator Machines. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 4181–4188, 2019. 9

[37] Fangda Gu, Armin Askari, and Laurent El Ghaoui. Fenchel Lifted Networks: A Lagrange Relaxation of Neural Network Training. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 3362–3371, 2020. 9

[38] Yoshua Bengio. How Auto-Encoders Could Provide Credit Assignment in Deep Networks via Target Propagation. *arXiv preprint arXiv:1407.7906*, 2014. 9

[39] Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference Target Propagation. In *International Conference on Learning Representations (ICLR)*, page 498–515, 2015. 9

[40] Alexander Meulemans, Francesco Carzaniga, Johan Suykens, João Sacramento, and Benjamin F. Grewe. A Theoretical Framework for Target Propagation. In *Neural Information Processing Systems (NeurIPS)*, pages 20024–20036, 2020. 9

[41] Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled Neural Interfaces using Synthetic Gradients. In *International Conference on Machine Learning (ICML)*, pages 1627–1635, 2017. 9

[42] Wojciech Marian Czarnecki, Grzegorz Świrszcz, Max Jaderberg, Simon Osindero, Oriol Vinyals, and Koray Kavukcuoglu. Understanding Synthetic Gradients and Decoupled Neural Interfaces. In *International Conference on Machine Learning (ICML)*, pages 904–912, 2017. 9

[43] Benjamin James Lansdell, Prashanth Ravi Prakash, and Konrad Paul Kording. Learning to solve the credit assignment problem. In *International Conference on Learning Representations (ICLR)*, 2020. 9

[44] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *International Conference on Machine Learning (ICML)*, pages 1126–1135, 2017. 9

[45] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-Learning with Implicit Gradients. In *Neural Information Processing Systems (NeurIPS)*, 2019. 9

[46] Xin-Yu Zhang, Taihong Xiao, Haolin Jia, Ming-Ming Cheng, and Ming-Hsuan Yang. Semi-Supervised Learning with Meta-Gradient. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 73–81, 2021. 9

[47] J Frédéric Bonnans and Alexander Shapiro. *Perturbation analysis of optimization problems*. Springer Science & Business Media, 2013. 9

[48] Justin Domke. Generic Methods for Optimization-Based Modeling. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 318–326, 2012. 9

[49] Dougal Maclaurin, David Duvenaud, and Ryan Adams. Gradient-based Hyperparameter Optimization through Reversible Learning. In *International Conference on Machine Learning (ICML)*, pages 2113–2122, 2015. 9

[50] Jelena Luketina, Mathias Berglund, Klaus Greff, and Tapani Raiko. Scalable Gradient-Based Tuning of Continuous Regularization Hyperparameters. In *International Conference on Machine Learning (ICML)*, pages 2952–2960, 2016. 9

[51] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable Architecture Search. *International Conference on Learning Representations (ICLR)*, 2018. 9

[52] Zhengyang Geng, Meng-Hao Guo, Hongxu Chen, Xia Li, Ke Wei, and Zhouchen Lin. Is Attention Better Than Matrix Decomposition? In *International Conference on Learning Representations (ICLR)*, 2021. 9

[53] Fabian Pedregosa. Hyperparameter Optimization with Approximate Gradient. In *International Conference on Machine Learning (ICML)*, pages 737–746, 2016. 9

Checklist

1. For all authors...
   - (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   - (b) Did you describe the limitations of your work? [No]
   - (c) Did you discuss any potential negative societal impacts of your work? [N/A] Not applicable.
   - (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   - (a) Did you state the full set of assumptions of all theoretical results? [Yes] See the assumptions in Theorem 1 to 3.
   - (b) Did you include complete proofs of all theoretical results? [Yes] See the appendix.
3. If you ran experiments...
   - (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   - (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   - (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   - (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   - (a) If your work uses existing assets, did you cite the creators? [Yes] See Sec. 4.
   - (b) Did you mention the license of the assets? [Yes]
   - (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See supplemental materials.
   - (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   - (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   - (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   - (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   - (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]