iDARTS: Differentiable Architecture Search with Stochastic Implicit Gradients

Miao Zhang [1 2] Steven Su [2] Shirui Pan [1] Xiaojun Chang [1] Ehsan Abbasnejad [3] Reza Haffari [1]

Abstract

Differentiable ARchiTecture Search (DARTS) has recently become the mainstream of neural architecture search (NAS) due to its efficiency and simplicity. With a gradient-based bi-level optimization, DARTS alternately optimizes the inner model weights and the outer architecture parameter in a weight-sharing supernet. A key challenge to the scalability and quality of the learned architectures is the need for differentiating through the inner-loop optimisation. While much has been discussed about several potentially fatal factors in DARTS, the architecture gradient, a.k.a. hypergradient, has received less attention. In this paper, we tackle the hypergradient computation in DARTS based on the implicit function theorem, making it only depends on the obtained solution to the inner-loop optimization and agnostic to the optimization path. To further reduce the computational requirements, we formulate a stochastic hypergradient approximation for differentiable NAS, and theoretically show that the architecture optimization with the proposed method, named iDARTS, is expected to converge to a stationary point. Comprehensive experiments on two NAS benchmark search spaces and the common NAS search space verify the effectiveness of our proposed method. It leads to architectures outperforming, with large margins, those learned by the baseline methods.

1. Introduction

Neural Architecture Search (NAS) is an efficient and effective method on automating the process of neural network design, with achieving remarkable success on image recognition (Tan & Le, 2019; Li et al., 2021b; 2020), language modeling (Jiang et al., 2019), and other deep learning applications (Ren et al., 2020; Cheng et al., 2020; Chen et al., 2019b; Hu et al., 2021; Zhu et al., 2021; Ren et al., 2021). The early NAS frameworks are devised via reinforcement learning (RL) (Pham et al., 2018) or evolutionary algorithm (EA) (Real et al., 2019) to directly search on the discrete space. To further improve the efficiency, a recently proposed *Differentiable ARchiTecture Search* (DARTS) (Liu et al., 2019) adopts the continuous relaxation to convert the operation selection problem into the continuous magnitude optimization for a set of candidate operations. By enabling the gradient descent for the architecture optimization, DARTS significantly reduces the search cost to several GPU hours (Liu et al., 2019; Xu et al., 2020; Dong & Yang, 2019a).

Despite its efficiency, more current works observe that DARTS is somewhat unreliable (Zela et al., 2020a; Chen & Hsieh, 2020; Li & Talwalkar, 2019; Sciuto et al., 2019; Zhang et al., 2020c; Li et al., 2021a; Zhang et al., 2020b) since it does not consistently yield excellent solutions, performing even worse than random search in some cases. Zela et al. (2020a) attribute the failure of DARTS to its supernet training, with empirically observing that the instability of DARTS is highly correlated to the dominant eigenvalue of the Hessian matrix of the validation loss with respect to architecture parameters. On the other hand, Wang et al. (2021a) turn to the magnitude-based architecture selection process, who empirically and theoretically show the magnitude of architecture parameters does not necessarily indicate how much the operation contributes to the supernet's performance. Chen & Hsieh (2020) observe a precipitous validation loss landscape with respect to architecture parameters, which leads to a dramatic performance drop when discretizing the final architecture for the operation selection. Accordingly, they propose a perturbation based regularization to smooth the loss landscape and improve the stability.

While there are many variants on improving the DARTS from various aspects, limited research attention has been paid to the approximation of the architecture parameter gradient, which is also called the outer-loop gradient or hypergradient. To fill the gap, this paper focuses on the hypergradient calculation in the differentiable NAS. The main contribution of this work is the development of the differentiable architecture search with stochastic implicit gradients (iDARTS). Specifically, we first revisit the DARTS from

[1]Faculty of Information Technology, Monash University, Australia [2]Faculty of Engineering and Information Technology, University of Technology Sydney, Australia [3]Australian Institute for Machine Learning, University of Adelaide, Australia. Correspondence to: Shirui Pan <Shirui.Pan@monash.edu>.

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

the bi-level optimization perspective and utilize the implicit function theorem (IFT) (Bengio, 2000; Lorraine et al., 2020), instead of the one-step unroll learning paradigm adopted by DARTS, to calculate the architecture parameter gradient. This IFT based hypergradient depends only on the obtained solution to the inner-loop optimization weights rather than the path taken, thus making the proposed method memory efficient and practical with numerous inner optimization steps. Then, to avoid calculating the inverse of the Hessian matrix with respect to the model weights, we utilize the Neumann series (Lorraine et al., 2020) to approximate this inverse and propose an approximated hypergradient for DARTS accordingly. After that, we devise a stochastic approximated hypergradient to relieve the computational burden further, making the proposed method applicable to the differentiable NAS. We theoretically demonstrate that, under some mild assumptions (Ghadimi & Wang, 2018; Couellan & Wang, 2016; Grazzi et al., 2020b) on the inner and outer loss functions, the proposed method is expected to converge to a stationary point with small enough learning rates. Finally, we verify the effectiveness of the proposed approach on two NAS benchmark datasets and the common DARTS search space.

We make the following contributions:

- This paper deepens our understanding of the hypergradient calculation in the differentiable NAS. We reformulated the hypergradient in the differentiable NAS with the implicit function theorem (IFT), which can thus gracefully handle many inner optimization steps without increasing the memory requirement.
- To relieve the heavy computational burdens, we consider a Neumann-approximation for the IFT based differentiable NAS. Further, to make the implicit hypergradient practical for differentiable NAS, we formulate a stochastic hypergradient with the Neumann-approximation.
- We provide a theoretical analysis of the proposed method and demonstrate that the proposed method is expected to converge to a stationary point when applied to differentiable NAS. Extensive experiments verify the effectiveness of the proposed method which significantly improves the performance of the differentiable NAS baseline on the NAS-Bench-1Shot1 and the NAS-Bench-201 benchmark datasets and the common DARTS search space.

2. Preliminaries: DARTS and Bi-level Optimization

Existing differentiable NAS methods mostly leverage the weight sharing and continuous relaxation to enable the gradient descent for the discrete architecture search, significantly improving the search efficiency. DARTS (Liu et al., 2019) is one of the most representative differentiable NAS methods, which utilizes the continuous relaxation to convert the discrete operation selection into the magnitude optimization for a set of candidate operations. Typically, NAS searches for cells to stack the full architecture, where a cell structure is represented as a directed acyclic graph (DAG) with N nodes. NAS aims to determine the operations and corresponding connections for each node, while DARTS applies a **softmax** function to calculate the magnitude of each operation, transforming the operation selection into a continuous magnitude optimization problem:

$$\mathbf{X}_n = \sum_{0 \le s < n} \sum_{o=1}^{|\mathcal{O}|} \bar{\alpha}_o^{(s,n)} o(\mathbf{X}_s), \qquad \bar{\alpha}_o^{(s,n)} = \frac{\exp(\alpha_o^{s,n})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{s,n})},$$

where $\mathbf{X}_n$ is the output of node n, $\mathcal{O}$ contains all candidate operations, and the output of each node is the weighted sum of its previous nodes' outputs affiliated with all possible operations. In this way, DARTS transforms the discrete architecture search into optimizing the continuous magnitude $\hat{\alpha}_o^{s,n}$, enabling gradient descent for the architecture optimization. A discrete architecture is obtained by applying an **argmax** function to the magnitude matrix after the differentiable architecture optimization.

The optimization in DARTS is based on the bi-level optimization formulation (Colson et al., 2007; Liu et al., 2019):

$$\begin{aligned} \min_{\alpha} \quad & \mathcal{L}_{val}(w^*(\alpha), \alpha) \\ \text{s.t.} \quad & w^*(\alpha) = \text{argmin}_w \, \mathcal{L}_{train}(w, \alpha), \end{aligned} \tag{1}$$

where α is the continuous architecture representation and w is the supernet weights. We indicate the $\mathcal{L}_{val}$ as $\mathcal{L}_2$ and the $\mathcal{L}_{train}$ as $\mathcal{L}_1$ in the remaining text for convenience. The nested formulation in DARTS is the same as the gradient-based hyperparameter optimization with bi-level optimization (Franceschi et al., 2018; Maclaurin et al., 2015; Pedregosa, 2016), where the inner-loop is to train the network parameter w and the outer-loop is to optimize the architecture parameter α. The gradient of the outer-loop for DARTS is then calculated as:

$$\nabla_\alpha \mathcal{L}_2 = \left(\frac{\partial \mathcal{L}_2}{\partial \alpha} + \frac{\partial \mathcal{L}_2}{\partial w} \frac{\partial w^*(\alpha)}{\partial \alpha}\right). \tag{2}$$

DARTS considers the one-step unroll learning paradigm (Liu et al., 2019; Rajeswaran et al., 2019) for the hypergradient calculation. This is done by taking a single step in optimising w instead of the optimal w^*.

Different from the majority of existing works that attributes the failure of DARTS to its supernet optimization (Zela et al., 2020a; Benyahia et al., 2019), or the final discretization with **argmax** (Chen & Hsieh, 2020; Wang et al., 2021b), this paper revisits DARTS from the perspective of the hypergradient calculation $\nabla_\alpha \mathcal{L}_2$. Rather than considering the one-step unroll learning paradigm (Liu et al., 2019; Finn et al., 2017), this paper utilizes the implicit function theorem (IFT) (Bengio, 2000; Lorraine et al., 2020) to reformulate the hypergradient calculation in DARTS. In the following subsection, we first recap the hypergradient calculation with different paradigms for DARTS.

3. Hypergradient: From Unrolling to iDARTS

One-step unrolled differentiation. The one-step unroll learning paradigm, as adopted by DARTS, is commonly used in the bi-level optimization based applications, including meta-learning (Finn et al., 2017), hyperparameter optimization (Luketina et al., 2016), generative adversarial networks (Metz et al., 2017), and neural architecture search (Liu et al., 2019), as it simplifies the hypergradient calculation and makes the bi-level optimization formulation practical for large-scale network learning. As described in the Section 2, the one-step unroll learning paradigm restricts the inner-loop optimization with only one step training. Differentiating through the inner learning procedure with one step $w^*(\alpha) = w - \gamma\nabla_w \mathcal{L}_1$, and obtaining $\frac{\partial w^*(\alpha)}{\partial \alpha} = -\gamma \frac{\partial^2 \mathcal{L}_1}{\partial \alpha \partial w}$, DARTS calculates the hypergradient as:

$$\nabla_\alpha \mathcal{L}_2^{DARTS} = \frac{\partial \mathcal{L}_2}{\partial \alpha} - \gamma \frac{\partial \mathcal{L}_2}{\partial w}\frac{\partial^2 \mathcal{L}_1}{\partial \alpha \partial w}, \tag{3}$$

where γ is the inner-loop learning rate for w.

Reverse-mode back-propagation. Another direction of computing hypergradient is the reverse-mode (Franceschi et al., 2017; Shaban et al., 2019), which trains the inner-loop with enough steps to reach the optimal points for the inner optimization. This paradigm assumes T-step is large enough to adapt $w(\alpha)$ to $w^*(\alpha)$ in the inner-loop. Defining Φ as a step of inner optimization that $w_t(\alpha) = \Phi(w_{t-1}, \alpha)$, and defining $Z_t = \nabla_\alpha w_t(\alpha)$, we have:

$$Z_t = A_t Z_{t-1} + B_t,$$

where $A_t = \frac{\partial \Phi(w_{t-1},\alpha)}{\partial w_{t-1}}$, and $B_t = \frac{\partial \Phi(w_{t-1},\alpha)}{\partial \alpha}$.

Then the hypergradient of DARTS with the reverse model could be formulated as:

$$\nabla_\alpha \mathcal{L}_2^{Reverse} = \frac{\partial \mathcal{L}_2}{\partial \alpha} + \frac{\partial \mathcal{L}_2}{\partial w_T}(\sum_{t=0}^{T} B_t A_{t+1}...A_T). \tag{4}$$

Although the reverse-mode bi-level optimization is easy to implement, the memory requirement linearly increases with the number of steps T (Franceschi et al., 2017) as it needs to store all intermediate gradients, making it impractical for deep networks. Rather than storing the gradients for all steps, a recent work (Shaban et al., 2019) only uses the last K-step ($K << T$) gradients to approximate the exact hypergradient, which is called the truncated back-propagation. Based on the K-step truncated back-propagation, the hypergradient for DARTS could be described as:

$$h_{T-K} = \frac{\partial \mathcal{L}_2}{\partial \alpha} + \frac{\partial \mathcal{L}_2}{\partial w_T} Z_T = \frac{\partial \mathcal{L}_2}{\partial \alpha} + \frac{\partial \mathcal{L}_2}{\partial w_T}(\sum_{t=T-K+1}^{T} B_t A_{t+1}...A_T). \tag{5}$$

The lemmas in (Shaban et al., 2019) show that h_{T-K} is a sufficient descent direction for the outer-loop optimization.

Lemma 1 *(Shaban et al., 2019). For all $K \geq 1$, with T large enough and γ small enough, h_{T-K} is a sufficient descent direction that, i.e. $h_{T-K}^\top \nabla_\alpha \mathcal{L}_2 \geq \Omega(\|\nabla_\alpha \mathcal{L}_2\|^2)$.*

iDARTS: Implicit gradients differentiation. Although h_{T-K} significantly decreases memory requirements, it still needs to store K-step gradients, making it impractical for differentiable NAS. In contrast, by utilizing the implicit function theorem (IFT), the hypergradient can be calculate without storing the intermediate gradients (Bengio, 2000; Lorraine et al., 2020). The IFT based hypergradient for DARTS could be formulated as the following lemma.

Lemma 2 *Implicit Function Theorem: Consider $\mathcal{L}_1$, $\mathcal{L}_2$, and $w^*(\alpha)$ as defined in Eq.(1), and with $\frac{\partial \mathcal{L}_1(w^*,\alpha)}{\partial w} = 0$, we have*

$$\nabla_\alpha \mathcal{L}_2 = \frac{\partial \mathcal{L}_2}{\partial \alpha} - \frac{\partial \mathcal{L}_2}{\partial w}\left[\frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^{-1}\frac{\partial^2 \mathcal{L}_1}{\partial \alpha \partial w}. \tag{6}$$

This is also called as implicit differentiation theorem (Lorraine et al., 2020). However, for a large neural network, it is hard to calculate the inverse of Hessian matrix in Eq.(6), and one common direction is to approximate this inverse. Compared with Eq.(6), the hypergradient of DARTS (Liu et al., 2019) in Eq.(3), which adopts the one-step unrolled differentiation, simply uses an identity to approximate the inverse $\left[\frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^{-1} = \gamma I$. This naive approximation is also adopted by (Luketina et al., 2016; Balaji et al., 2018; Nichol et al., 2018). In contrast, (Rajeswaran et al., 2019; Pedregosa, 2016) utilize the conjugate gradient (CG) to convert the approximation of the inverse to solving a linear system with δ-optimal solution, with applications to the hyperparameter optimization and meta-learning.

Recently, the Neumann series is introduced to approximate the inverse in the hyperparameter optimization (Lorraine et al., 2020) for modern and deep neural networks since it is a more stable alternative to CG and useful in stochastic settings. This paper thus adopts the Neumann series for the inverse approximation and proposes an approximated hypergradient for DARTS accordingly. A stochastic approximated hypergradient is further devised to fit with differentiable NAS and relieve the computational burden, which is called Differentiable Architecture Search with Stochastic Implicit Gradients (**iDARTS**). We theoretically show the proposed method converges in expectation to a stationary point for the differentiable architecture search with small enough learning rates. A detailed description and analysis of **iDARTS** follow in the next section.

4. Stochastic Approximations in iDARTS

As described, our **iDARTS** utilizes the Neumann series to approximate the inverse of the Hessian matrix for the hypergradient calculation in the IFT-based bi-level optimization of NAS. We further consider a stochastic setting where the Neumann approximation is computed based on minibatch samples, instead of the full dataset, enabling scalability to large datasets, similar to standard-practice in deep learning.

This section starts by analyzing the bound of the proposed hypergradient approximation, and then shows the convergence property of the proposed stochastic approximated hypergradient for differentiable NAS.

Before our analysis, we give the following common assumptions in the bi-level optimization.[1]

Assumption 1 *For the outer-loop function $\mathcal{L}_2$:*

1. *For any w and α, $\mathcal{L}_2(w,\cdot)$ and $\mathcal{L}_2(\cdot,\alpha)$ are bounded below.*
2. *For any w and α, $\mathcal{L}_2(w,\cdot)$ and $\mathcal{L}_2(\cdot,\alpha)$ are Lipschitz continuous with constants $L_2^w > 0$ and $L_2^\alpha > 0$.*
3. *For any w and α, $\nabla_w\mathcal{L}_2(w,\cdot)$ and $\nabla_\alpha\mathcal{L}_2(\cdot,\alpha)$ are Lipschitz continuous with constants $L_2^{\nabla_w} > 0$ and $L_2^{\nabla_\alpha} > 0$ with respect to w and α.*

Assumption 2 *For the inner-loop function $\mathcal{L}_1$*

1. *$\nabla_w\mathcal{L}_1$ is Lipschitz continuous with respect to w with constant $L_1^{\nabla_w} > 0$.*
2. *The function $w : \alpha \to w(\alpha)$ is Lipschitz continuous with constant $L_w > 0$, and has Lipschitz gradient with constant $L_{\nabla_\alpha w} > 0$.*
3. *$\|\nabla^2_{w\alpha}\mathcal{L}_1\|$ is bounded that $\|\nabla^2_{w\alpha}\mathcal{L}_1\| \leq C_{\mathcal{L}_1^{w\alpha}}$ for some constant $C_{\mathcal{L}_1^{w\alpha}} > 0$.*

4.1. Hypergradient based on Neumann Approximation

In this subsection, we describe how to use the Neumann series to reformulate the hypergradient in DARTS.

Lemma 3 *Neumann series (Lorraine et al., 2020): With a matrix A that $\|I-A\| < 1$, $A^{-1} = \sum_{k=0}^{\infty}(I-A)^k$.*

[1] Similar assumptions are also considered in (Couellan & Wang, 2016; Ghadimi & Wang, 2018; Grazzi et al., 2020b;a).

Based on Lemma 3, the Eq. (6) for the IFT based DARTS is formulated by Eq. (7) as described in Corollary 1.

Corollary 1 *With small enough learning rate $\gamma < \frac{1}{L_1^{\nabla_w}}$, the hypergradient in DARTS can be formulated as:*

$$\begin{aligned}\nabla_\alpha\mathcal{L}_2 &= \frac{\partial\mathcal{L}_2}{\partial\alpha} - \frac{\partial\mathcal{L}_2}{\partial w}\left[\frac{\partial^2\mathcal{L}_1}{\partial w\partial w}\right]^{-1}\frac{\partial^2\mathcal{L}_1}{\partial\alpha\partial w} \\ &= \frac{\partial\mathcal{L}_2}{\partial\alpha} - \gamma\frac{\partial\mathcal{L}_2}{\partial w}\sum_{j=0}^{\infty}\left[I-\gamma\frac{\partial^2\mathcal{L}_1}{\partial w\partial w}\right]^j\frac{\partial^2\mathcal{L}_1}{\partial\alpha\partial w}.\end{aligned} \tag{7}$$

As shown in the Corollary 1, the approximated hypergradient for DARTS, denoted by $\nabla_\alpha\tilde{\mathcal{L}}_2$ could be obtained by only considering the first K terms of Neumann approximation without calculating the inverse of Hessian (Shaban et al., 2019; Lorraine et al., 2020) as,

$$\nabla_\alpha\tilde{\mathcal{L}}_2 = \frac{\partial\mathcal{L}_2}{\partial\alpha} - \gamma\frac{\partial\mathcal{L}_2}{\partial w}\sum_{k=0}^{K}\left[I-\gamma\frac{\partial^2\mathcal{L}_1}{\partial w\partial w}\right]^k\frac{\partial^2\mathcal{L}_1}{\partial\alpha\partial w}. \tag{8}$$

As shown, we could observe the relationship between the proposed $\nabla_\alpha\tilde{\mathcal{L}}_2$ and the hypergradient of DARTS in Eq(3), which is the same as $\nabla_\alpha\tilde{\mathcal{L}}_2$ when $K=0$. In the following theorem, we give the error bound between our approximated hypergradient $\nabla_\alpha\tilde{\mathcal{L}}_2$ and the exact hypergradient $\nabla_\alpha\mathcal{L}_2$.

Theorem 1 *Suppose the inner optimization function $\mathcal{L}_1$ is twice differentiable and is μ-strongly convex with w around $w^*(\alpha)$. The error between the approximated gradient $\nabla_\alpha\tilde{\mathcal{L}}_2$ and $\nabla_\alpha\mathcal{L}_2$ in DARTS is bounded with $\left\|\nabla_\alpha\mathcal{L}_2 - \nabla_\alpha\tilde{\mathcal{L}}_2\right\| \leqslant C_{\mathcal{L}_1^{w\alpha}}C_{\mathcal{L}_2^{w}}\frac{1}{\mu}(1-\gamma\mu)^{K+1}$.*

Theorem 1 states that the approximated hypergradient approaches to the exact hypergradient as K increases. As described, the form of $\nabla_\alpha\tilde{\mathcal{L}}_2$ is similar to the K-step truncated back-propagation in Eq. (5), while the memory consumption of our $\nabla_\alpha\tilde{\mathcal{L}}_2$ is only $\frac{1}{K}$ of the memory needed to compute h_{T-K}, as we only store the gradients of the final solution w^*. In the following corollary, we describe the connection between the proposed approximated hypergradient $\nabla_\alpha\tilde{\mathcal{L}}_2$ and the approximation based on the truncated back-propagation h_{T-K} (Shaban et al., 2019).

Corollary 2 *When we assume w_t has converged to a stationary point w^* in the last K steps, the proposed $\nabla_\alpha\tilde{\mathcal{L}}_2$ is the same as the truncated back-propagation h_{T-K}.*

4.2. Stochastic Approximation of Hypergradient

The Lemma 1 and Corollary 2 show that the approximated hypergradient $\nabla_\alpha\tilde{\mathcal{L}}_2$ has the potential to be a sufficient descent direction. However, it is not easy to calculate the implicit gradients for DARTS based on Eq. (8) as it needs to deal with large-scale datasets in which the loss functions are large sums of error terms:

$$\mathcal{L}_2 = \frac{1}{R}\sum_{i=1}^{R}\mathcal{L}_2^i; \qquad \mathcal{L}_1 = \frac{1}{J}\sum_{j=1}^{J}\mathcal{L}_1^j,$$

where J is the number of minibatches of the training dataset $\mathcal{D}_{train}$ for the inner supernet training $\mathcal{L}_1$, and R is the number of minibatches of the validation dataset $\mathcal{D}_{val}$ for the outer architecture optimization $\mathcal{L}_2$. It is apparently challenging to calculate the gradient based on the full dataset in each step. We therefore utilize the stochastic gradient based on individual minibatches in practice. That is, we consider the following stochastic approximated hypergradient,

$$\nabla_\alpha \hat{\mathcal{L}}_2^i(w^j(\alpha),\alpha) = \frac{\partial \mathcal{L}_2^i}{\partial \alpha} - \gamma \frac{\partial \mathcal{L}_2^i}{\partial w}\sum_{k=0}^{K}\left[I - \gamma\frac{\partial^2 \mathcal{L}_1^j}{\partial w \partial w}\right]^k \frac{\partial^2 \mathcal{L}_1^j}{\partial \alpha \partial w}. \qquad (9)$$

where $\mathcal{L}_2^i$ and $\mathcal{L}_1^j$ correspond to loss functions calculated by randomly sampled minibatches i and j from $\mathcal{D}_{val}$ and $\mathcal{D}_{train}$, respectively. This expression can be computed using the Hessian-vector product technique without explicitly computing the Hessian matrix (see Appendix B). Before analyzing the convergence of the proposed $\nabla_\alpha \hat{\mathcal{L}}_2(w^j(\alpha),\alpha)$, we give the following lemma to show function $\mathcal{L}_2 : \alpha \to \mathcal{L}_2(w,\alpha)$ is differentiable with a Lipschitz continuous gradient (Couellan & Wang, 2016).

Lemma 4 *Based on the Assumption 1 and 2, we have the function $\mathcal{L}_2 : \alpha \to \mathcal{L}_2(w,\alpha)$ is differentiable with Lipschitz continuous gradient and Lipschitz constant $L_{\nabla_\alpha \mathcal{L}_2} = L_2^{\nabla_\alpha} + L_2^{\nabla_w} L_w^2 + L_2^w L_{\nabla_\alpha w}$.*

Then we state and prove the main convergence theorem for the proposed stochastic approximated hypergradient $\nabla_\alpha \hat{\mathcal{L}}_2^i(w^j(\alpha),\alpha)$ for the differentiable NAS.

Theorem 2 *Based on several assumptions, we could prove the convergence of the proposed stochastic approximated hypergradient for differentiable NAS. Suppose that:*

1. *All assumptions in Assumption 1 and 2 and Corollary 1 are satisfied;*
2. $\exists D > 0$ *such that* $E\left[\|\varepsilon\|^2\right] \le D\,\|\nabla_\alpha \mathcal{L}_2\|^2$;
3. $\forall i > 0$, γ_{α_i} *satisfies* $\sum_{i=1}^{\infty}\gamma_{\alpha_i} = \infty$ *and* $\sum_{i=1}^{\infty}\gamma_{\alpha_i}^2 < \infty$.
4. *The inner function $\mathcal{L}_1$ has the special structure: $\mathcal{L}_1^j(w,\alpha) = h(w,\alpha) + h_j(w,\alpha), \quad \forall j \in 1,...,J$, that h_j is a linear function with respect to w and α.*

With small enough learning rate γ_α for the architecture optimization, the proposed stochastic hypergradient based algorithm converges in expectation to a stationary point, i.e. $\lim_{m\to\infty} E\left[\left\|\nabla_\alpha \hat{\mathcal{L}}_2^i(w^j(\alpha_m),\alpha_m)\right\|\right] = 0.$

Algorithm 1 iDARTS

```
Input: $\mathcal{D}_{train}$ and $\mathcal{D}_{val}$. Initialized supernet weights $w$ and operations magnitude $\alpha_\theta$.
   while not converged do
2:     ⋆ Sample batches from $\mathcal{D}_{train}$. Update supernet weights $w$ based on cross-entropy loss with $T$ steps.
       ⋆ Get the Hessian matrix $\frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}$.
4:     ⋆ Sample batch from $\mathcal{D}_{val}$. Calculate hypergradient $\nabla_\alpha \hat{\mathcal{L}}_2^i(w^j(\alpha),\alpha)$ based on Eq.(9), and update $\alpha$ with $\alpha \leftarrow \alpha - \gamma_\alpha \nabla_\alpha \hat{\mathcal{L}}_2^i(w^j(\alpha),\alpha)$.
   end while
6: Obtain $\alpha^*$ through argmax.
```

The ε is defined as the noise term between the stochastic gradient $\nabla_\alpha \mathcal{L}_2^i(w^j(\alpha),\alpha)$ and the true gradient $\nabla_\alpha \mathcal{L}_2$ as:

$$\varepsilon_{i,j} = \nabla_\alpha \mathcal{L}_2 - \nabla_\alpha \mathcal{L}_2^i(w^j(\alpha),\alpha).$$

where $\nabla_\alpha \mathcal{L}_2^i(w^j(\alpha),\alpha)$ is the non-approximate version of Eq. (9) when $K \to \infty$.

Theorem 2 shows that the proposed stochastic approximated hypergradient is also a sufficient descent direction, which leads the differentiable NAS converges to a stationary point. The conditions 2-4 in Theorem 2 are common assumptions in analyzing the stochastic bi-level gradient methods (Couellan & Wang, 2016; Ghadimi & Wang, 2018; Grazzi et al., 2020b;a). We assume that $\mathcal{L}_1$ in Eq. (7) is μ-strongly convex with w around w^*, which can be made possible by appropriate choice of learning rates (Rajeswaran et al., 2019; Shaban et al., 2019). Another key assumption in our convergence analysis is the Lipshitz differentiable assumptions for $\mathcal{L}_1$ and $\mathcal{L}_2$ in Assumption 1 and 2, which also received considerable attention in recent optimization and deep learning literature (Jin et al., 2017; Rajeswaran et al., 2019; Lorraine et al., 2020; Mackay et al., 2018; Grazzi et al., 2020a).

4.3. Differentiable Architecture Search with Stochastic Implicit Gradients

Different from DARTS that alternatively optimizes both α and w with only one step in each round, iDARTS is supposed to train the supernet with enough steps to make sure the $w(\alpha)$ is near $w^*(\alpha)$ before optimizing α. The framework of our iDARTS is sketched in Algorithm 1. Generally, it is impossible to consider a very large T for each round of supernet weights w optimization, as the computational cost increase linear with T. Fortunately, empirical experiments show that, with the weight sharing, the differentiable NAS can adapt $w(\alpha)$ to $w^*(\alpha)$ with a small T in the later phase of architecture search.

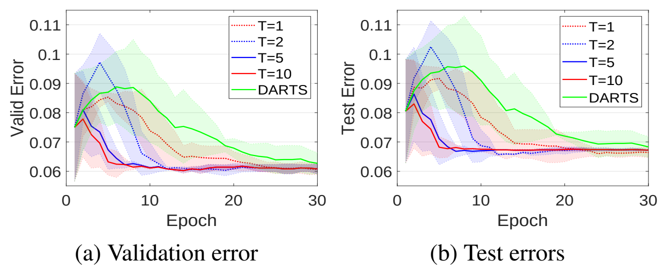


Figure 1. Validation and test errors of iDARTS with different T and DARTS on the search space 3 of NAS-Bench-1Shot1.

5. Experiments

In Section 4, we have theoretically shown that our iDARTS can asymptotically compute the exact hypergradient and lead to a convergence in expectation to a stationary point for the architecture optimization. In this section, we conduct a series of experiments to verify whether the iDARTS leads to better results in the differentiable NAS with realistic settings. We consider three different cases to analyze iDARTS, including two NAS benchmark datasets, NAS-Bench-1Shot1 (Zela et al., 2020b) and NAS-Bench-201 (Dong & Yang, 2020), and the common DARTS search space (Liu et al., 2019). We first analyze the iDARTS on the two NAS benchmark datasets, along with discussions of hyperparameter settings. Then we compare iDARTS with state-of-the-art NAS methods on the common DARTS search space.

5.1. Reproducible Comparison on NAS-Bench-1Shot1

To study the empirical performance of iDARTS, we run the iDARTS on the NAS-Bench-1Shot1 dataset with different *random seeds* to report its statistical results, and compare with the most closely related baseline DARTS (Liu et al., 2019). The NAS-Bench-1Shot1 is built from the NAS-Bench-101 benchmark dataset (Ying et al., 2019), through dividing all architectures in NAS-Bench-101 into 3 different unified cell-based search spaces. The architectures in each search space have the same number of nodes and connections, making the differentiable NAS could be directly applied to each search space. The three search spaces contain 6240, 29160, and 363648 architectures with the CIFAR-10 performance, respectively. We choose the third search space in NAS-Bench-1Shot1 to analyse iDARTS, since it is much more complicated than the remaining two search spaces and is a better case to identify the advantages of iDARTS.

Figure 1 plots the mean and standard deviation of the validation and test errors for iDARTS and DARTS, with tracking the performance during the architecture search on the NAS-Bench-1Shot1 dataset. As shown, our iDARTS with different T generally outperforms DARTS during the architecture search in terms of both validation and test error. More specifically, our iDARTS significantly outperforms the baseline in the early stage, demonstrating that our iDARTS could quickly find superior architectures and is more stable.

As described, one significant difference from DARTS is that iDARTS can conduct more than one training step in the inner-loop optimization. Figure 1 also analyzes the effects of the inner optimization steps T, plotting the performance of iDARTS with different T on the NAS-Bench-1Shot1. As shown, the inner optimization steps positively affect the performance of iDARTS, where increasing T helps iDARTS converge to excellent solutions faster. One underlying reason is that increasing T could adapt w to a local optimal w^*, thus helping iDRTS approximate the exact hypergradient more accurately. We should notice that the computational cost of iDARTS also increases with T, and our empirical findings suggest a $T = 5$ achieves an excellent compute and performance trade-off for iDARTS on NAS-Bench-1shot1. More interesting, iDARTS with $T = 1$ is similar as DARTS which both conduct the inner optimization with only one step, with the difference that iDARTS adopts the Neumann approximation while DARTS considers the unrolled differentiation. We could observe that iDARTS still outperforms DARTS by large margins in this case, showing the superiority of the proposed approximation over DARTS.

5.2. Reproducible Comparison on NAS-Bench-201

The NAS-Bench-201 dataset (Dong & Yang, 2020) is another popular NAS benchmark dataset to analyze differentiable NAS methods. The search space in NAS-Bench-201 contains four nodes with five associated operations, resulting in 15,625 cell candidates. The search space of NAS-Bench-201 is much simpler than NAS-Bench-1Shot1, while it contains the performance of CIFAR-100, CIFAR-100, and ImageNet for all architectures in this search space.

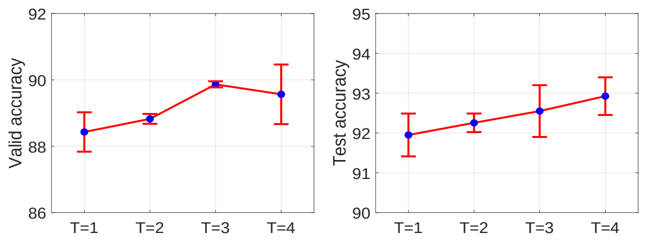
Table 1 summarizes the performance of iDARTS on NAS-Bench-201 compared with differentiable NAS baselines, where the statistical results are obtained from independent search experiments with different *random seeds*. As shown, our iDARTS achieved excellent results on the NAS-Bench-201 benchmark and significantly outperformed the DARTS baseline, with a 93.76%, 71.11%, and 41.44% test accuracy on CIFAR-10, CIFAR-100, and ImageNet, respectively. As described in Section 4, iDARTS is built based on the DARTS framework, with only reformulating the hypergradient calculation. These results in Table 1 verified the effectiveness of our iDARTS, which outperforms DARTS by large margins.

Similar to the experiments in the NAS-Bench-1Shot1, we also analyze the importance of hyperparameter T in the NAS-Bench-201 dataset. Figure 2 (a) summaries the performance of iDARTS with different number of inner optimization steps T on the NAS-Bench-201. As demonstrated, the performance of iDARTS is sensitive to the hyperparameter

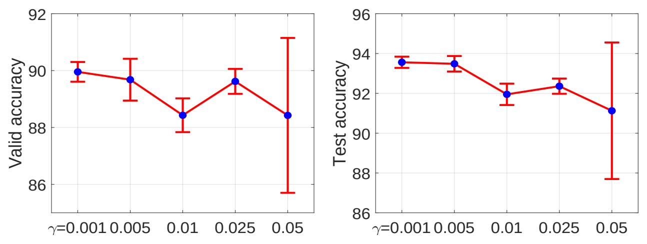
Table 1. Comparison results with NAS baselines on NAS-Bench-201.

Method	CIFAR-10		CIFAR-100		ImageNet-16-120	
	Valid(%)	Test(%)	Valid(%)	Test(%)	Valid(%)	Test(%)
ENAS (Pham et al., 2018)	37.51±3.19	53.89±0.58	13.37±2.35	13.96±2.33	15.06±1.95	14.84±2.10
RandomNAS (Li & Talwalkar, 2019)	80.42±3.58	84.07±3.61	52.12±5.55	52.31±5.77	27.22±3.24	26.28±3.09
SETN (Dong & Yang, 2019b)	84.04±0.28	87.64±0.00	58.86±0.06	59.05±0.24	33.06±0.02	32.52±0.21
GDAS (Dong & Yang, 2019a)	89.88±0.33	93.40±0.49	70.95±0.78	70.33±0.87	41.28±0.46	41.47±0.21
DARTS (Liu et al., 2019)	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00
iDARTS	89.86±0.60	93.58±0.32	70.57±0.24	70.83±0.48	40.38±0.593	40.89±0.68
optimal	91.61	94.37	74.49	73.51	46.77	47.31

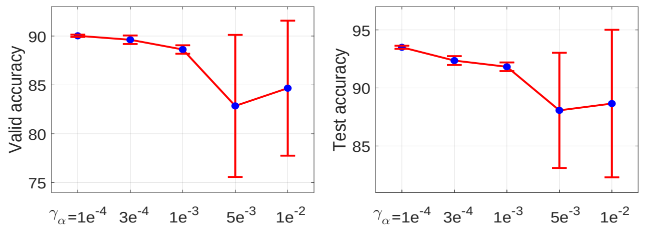
iDARTS's best single run achieves **93.76%**, **71.11%**, and **41.44%** test accuracy on CIFAR-10, CIFAR-100, and ImageNet, respectively.



(a) Validation and test performance with different T



(b) Validation and test performance with different γ



(c) Validation and test performance with different γ_α

Figure 2. Hyperparameter analysis of iDARTS on the NAS-Bench-201 benchmark dataset.

T, and a larger T helps iDARTS to achieve better results while also increases the computational time, which is also in line with the finding in the NAS-Bench-1Shot1. We empirically find that $T = 4$ is enough to achieve competitive results on NAS-Bench-201.

The hypergradient calculation of iDARTS is based on the Neumann approximation in Eq.(7), and one underlying condition is that the learning rates γ for the inner optimization should be small enough to make $\left\| I - \gamma \frac{\partial^2 \mathcal{L}_1}{\partial w \partial w} \right\| < 1$. We also conduct an ablation study to analyze how this hyperparameter affects our iDARTS, where Figure 2 (b) plots the performance of iDARTS with different learning rates γ for the inner optimization on the NAS-Bench-201. As shown, the performance of iDARTS is sensitive to γ, and a smaller γ is preferred, which also offers support for the Corollary 1 and Theorem 1.

During the analysis of the convergence of iDARTS, the learning rate γ_α plays a key role in the hypergradient approximation for the architecture optimization. Figure 2 (c) also summaries the performance of iDARTS with different initial learning rate γ_α on the NAS-Bench-201. As shown in Figure 2 (c), the performance of iDARTS is sensitive to γ_α, where a smaller γ_α is recommended, and a large γ_α is hardly able to converge to a stationary point. An underlying reason may lay in the proof of Theorem 2, that choosing a small enough γ_α guarantees that the iDARTS converges to a stationary point.

5.3. Experiments on DARTS Search Space

We also apply iDARTS to a convolutional architecture search in the common DARTS search space (Liu et al., 2019) to compare with the state-of-the-art NAS methods, where all experiment settings are following DARTS for fair comparisons. The search procedure needs to look for two different types of cells on CIFAR-10: normal cell α_{normal} and reduction cell α_{reduce}, to stack more cells to form the final structure for the architecture evaluation. The best-found cell on CIFAR-10 is then transferred to CIFAR-100 and ImageNet datasets to evaluate its transferability.

The comparison results with the state-of-the-art NAS methods are presented in Table 2, and Figure 3 demonstrates the best-found architectures by iDARTS. As shown in Table 2, iDARTS achieves a 2.37±0.03 % test error on CIFAR-10 (where the best single run is 2.35%), which is on par with the state-of-the-art NAS methods and outperforms the DARTS baseline by a large margin, again verifying the effectiveness of the proposed method.

Following DARTS experimental setting, the best-searched architectures on CIFAR-10 are then transferred to CIFAR-

Table 2. Comparison results with state-of-the-art weight-sharing NAS approaches.

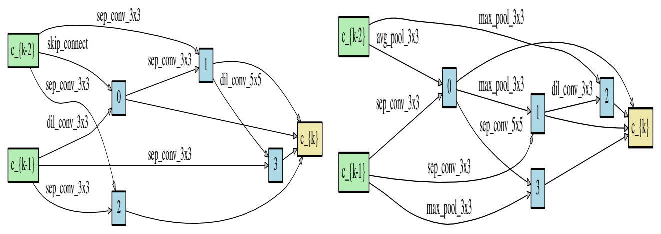
Method	Test Error (%)			Param	$+\times$	Architecture
	CIFAR-10	CIFAR-100	ImageNet	(M)	(M)	Optimization
NASNet-A (Zoph & Le, 2017)	2.65	17.81	26.0 / 8.4	3.3	564	RL
PNAS (Liu et al., 2018)	3.41 ± 0.09	17.63	25.8 / 8.1	3.2	588	SMBO
AmoebaNet-A (Real et al., 2019)	3.34 ± 0.06	-	25.5 / 8.0	3.2	555	EA
ENAS (Pham et al., 2018)	2.89	18.91	-	4.6	-	RL
EN^{2}AS (Zhang et al., 2020d)	2.61 ± 0.06	16.45	26.7 / 8.9	3.1	506	EA
RandomNAS (Li & Talwalkar, 2019)	2.85 ± 0.08	17.63	27.1	4.3	613	random
NSAS (Zhang et al., 2020a)	2.59 ± 0.06	17.56	25.5 / 8.2	3.1	506	random
PARSEC (Casale et al., 2019)	2.86 ± 0.06	-	26.3	3.6	509	gradient
SNAS (Xie et al., 2019)	2.85 ± 0.02	20.09	27.3 / 9.2	2.8	474	gradient
SETN (Dong & Yang, 2019b)	2.69	17.25	25.7 / 8.0	4.6	610	gradient
MdeNAS (Zheng et al., 2019)	2.55	17.61	25.5 / 7.9	3.6	506	gradient
GDAS (Dong & Yang, 2019a)	2.93	18.38	26.0 / 8.5	3.4	545	gradient
XNAS* (Nayman et al., 2019)	2.57 ± 0.09	16.34	24.7 / 7.5	3.7	600	gradient
PDARTS (Chen et al., 2019a)	2.50	16.63	24.4 / 7.4	3.4	557	gradient
PC-DARTS (Xu et al., 2020)	2.57 ± 0.07	17.11	25.1 / 7.8	3.6	586	gradient
DrNAS (Chen et al., 2020)	2.54 ± 0.03	16.30	24.2 / 7.3	4.0	644	gradient
DARTS (Liu et al., 2019)	2.76 ± 0.09	17.54	26.9 / 8.7	3.4	574	gradient
iDARTS	**2.37 ± 0.03**	**16.02**	**24.3 / 7.3**	3.8	595	gradient

"*" indicates the results reproduced based on the best-reported cell structures with a common experimental setting (Liu et al., 2019). "Param" is the model size when applied on CIFAR-10, while "$+\times$" is calculated based on the ImageNet dataset.

100 and ImageNet to evaluate the transferability. The evaluation setting for CIFAR-100 is the same as CIFAR-10. In the ImageNet dataset, the experiment setting is slightly different from CIFAR-10 in that only 14 cells are stacked, and the number of initial channels is changed to 48. We also follow the mobile setting in (Liu et al., 2019) to restrict the number of multiply-add operations ("$+\times$") to be less than 600M on the ImageNet. The comparison results with state-of-the-art differentiable NAS approaches on CIFAR-100 and ImageNet are demonstrated in Table 2. As shown, iDARTS delivers a competitive result with 16.02% test error on the CIFAR-100 dataset, which is a state-of-the-art performance and outperforms peer algorithms by a large margin. On the ImageNet dataset, the best-discovered architecture by our iDARTS also achieves a competitive result with 24.4 / 7.3 % top1 / top5 test error, outperforming or on par with all peer algorithms. Please note that, although DrNAS achieved outstanding performance on ImageNet, the number of multiply-add operations of its searched model is much over 600 M, violating the mobile-setting.

6. Conclusion

This paper opens up a promising research direction for NAS by focusing on the hypergradient approximation in the differentiable NAS. We introduced the implicit function theorem (IFT) to reformulate the hypergradient calculation in the differentiable NAS, making it practical with numerous inner optimization steps. To avoid calculating the inverse of the Hessian matrix, we utilized the Neumann series to approximate the inverse, and further devised a stochastic approximated hypergradient to relieve the computational cost. We theoretically analyzed the convergence and proved that the proposed method, called iDARTS, is expected to converge to a stationary point when applied to a differentiable NAS. We based our framework on DARTS and performed extensive experimental results that verified the proposed framework's effectiveness. While we only considered the proposed stochastic approximated hypergradient for differentiable NAS, iDARTS can in principle be used with a variety of bi-level optimization applications, including in meta-learning and hyperparameter optimization, opening up several interesting avenues for future research.



(a) normal cell (b) reduction cell

Figure 3. The best cells discovered by iDARTS on the DARTS search space.

Acknowledgement

This study was supported in part by the Australian Research Council (ARC) under a Discovery Early Career Researcher Award (DECRA) No. DE190100626 and DARPA's Learning with Less Labeling (LwLL) program under agreement FA8750-19-2-0501.

References

Balaji, Y., Sankaranarayanan, S., and Chellappa, R. Metareg: Towards domain generalization using meta-regularization. In *Advances in Neural Information Processing Systems*, pp. 998–1008, 2018.

Bengio, Y. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8):1889–1900, 2000.

Benyahia, Y., Yu, K., Smires, K. B., Jaggi, M., Davison, A. C., Salzmann, M., and Musat, C. Overcoming multi-model forgetting. In *International Conference on Machine Learning*, pp. 594–603, 2019.

Casale, F. P., Gordon, J., and Fusi, N. Probabilistic neural architecture search. *arXiv preprint arXiv:1902.05116*, 2019.

Chen, X. and Hsieh, C.-J. Stabilizing differentiable architecture search via perturbation-based regularization. In *ICML*, 2020.

Chen, X., Xie, L., Wu, J., and Tian, Q. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1294–1303, 2019a.

Chen, X., Wang, R., Cheng, M., Tang, X., and Hsieh, C.-J. Drnas: Dirichlet neural architecture search. *arXiv preprint arXiv:2006.10355*, 2020.

Chen, Y., Yang, T., Zhang, X., Meng, G., Xiao, X., and Sun, J. Detnas: Backbone search for object detection. In *Advances in Neural Information Processing Systems*, pp. 6642–6652, 2019b.

Cheng, X., Zhong, Y., Harandi, M., Dai, Y., Chang, X., Drummond, T., Li, H., and Ge, Z. Hierarchical neural architecture search for deep stereo matching. In *NeurIPS*, 2020.

Colson, B., Marcotte, P., and Savard, G. An overview of bilevel optimization. *Annals of operations research*, 153(1):235–256, 2007.

Couellan, N. and Wang, W. On the convergence of stochastic bi-level gradient methods. *Optimization*, 2016.

Dong, X. and Yang, Y. Searching for a robust neural architecture in four gpu hours. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2019a.

Dong, X. and Yang, Y. One-shot neural architecture search via self-evaluated template network. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3681–3690, 2019b.

Dong, X. and Yang, Y. Nas-bench-201: Extending the scope of reproducible neural architecture search. *ICLR*, 2020.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135, 2017.

Franceschi, L., Donini, M., Frasconi, P., and Pontil, M. Forward and reverse gradient-based hyperparameter optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1165–1173, 2017.

Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., and Pontil, M. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, pp. 1568–1577. PMLR, 2018.

Ghadimi, S. and Wang, M. Approximation methods for bilevel programming. *arXiv preprint arXiv:1802.02246*, 2018.

Grazzi, R., Franceschi, L., Pontil, M., and Salzo, S. On the iteration complexity of hypergradient computation. In *International Conference on Machine Learning*, pp. 3748–3758. PMLR, 2020a.

Grazzi, R., Pontil, M., and Salzo, S. Convergence properties of stochastic hypergradients. *arXiv preprint arXiv:2011.07122*, 2020b.

Griewank, A. Some bounds on the complexity of gradients, jacobians, and hessians. In *Complexity in numerical optimization*, pp. 128–162. World Scientific, 1993.

Griewank, A. and Walther, A. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

Hu, S., Zhu, F., Chang, X., and Liang, X. Updet: Universal multi-agent reinforcement learning via policy decoupling with transformers. *CoRR*, abs/2101.08001, 2021.

Jiang, Y., Hu, C., Xiao, T., Zhang, C., and Zhu, J. Improved differentiable architecture search for language modeling and named entity recognition. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3576–3581, 2019.

Jin, C., Ge, R., Netrapalli, P., Kakade, S. M., and Jordan, M. I. How to escape saddle points efficiently. In *International Conference on Machine Learning*, pp. 1724–1732. PMLR, 2017.

Li, C., Peng, J., Yuan, L., Wang, G., Liang, X., Lin, L., and Chang, X. Block-wisely supervised neural architecture search with knowledge distillation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 1986–1995, 2020.

Li, C., Tang, T., Wang, G., Peng, J., Wang, B., Liang, X., and Chang, X. Bossnas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search. *CoRR*, abs/2103.12424, 2021a.

Li, C., Wang, G., Wang, B., Liang, X., Li, Z., and Chang, X. Dynamic slimmable network. In *CVPR*, 2021b.

Li, L. and Talwalkar, A. Random search and reproducibility for neural architecture search. *arXiv preprint arXiv:1902.07638*, 2019.

Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., and Murphy, K. Progressive neural architecture search. In *ECCV*, 2018.

Liu, H., Simonyan, K., and Yang, Y. Darts: Differentiable architecture search. *ICLR*, 2019.

Lorraine, J., Vicol, P., and Duvenaud, D. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, pp. 1540–1552. PMLR, 2020.

Luketina, J., Berglund, M., Greff, K., and Raiko, T. Scalable gradient-based tuning of continuous regularization hyperparameters. In *International conference on machine learning*, pp. 2952–2960, 2016.

Mackay, M., Vicol, P., Lorraine, J., Duvenaud, D., and Grosse, R. Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. In *International Conference on Learning Representations*, 2018.

Maclaurin, D., Duvenaud, D., and Adams, R. Gradient-based hyperparameter optimization through reversible learning. In *International conference on machine learning*, pp. 2113–2122. PMLR, 2015.

Metz, L., Poole, B., Pfau, D., and Sohl-Dickstein, J. Unrolled generative adversarial networks. In *ICLR*, 2017.

Nayman, N., Noy, A., Ridnik, T., Friedman, I., Jin, R., and Zelnik, L. Xnas: Neural architecture search with expert advice. In *Advances in Neural Information Processing Systems*, pp. 1975–1985, 2019.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Pedregosa, F. Hyperparameter optimization with approximate gradient. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*, pp. 737–746, 2016.

Pham, H., Guan, M., Zoph, B., Le, Q., and Dean, J. Efficient neural architecture search via parameter sharing. In *International Conference on Machine Learning*, pp. 4092–4101, 2018.

Rajeswaran, A., Finn, C., Kakade, S. M., and Levine, S. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, pp. 113–124, 2019.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. Regularized evolution for image classifier architecture search. *AAAI*, 2019.

Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Chen, X., and Wang, X. A comprehensive survey of neural architecture search: Challenges and solutions. *arXiv preprint arXiv:2006.02903*, 2020.

Ren, P., Xiao, G., Chang, X., Xiao, Y., Li, Z., and Chen, X. NAS-TC: neural architecture search on temporal convolutions for complex action recognition. *CoRR*, abs/2104.01110, 2021.

Sciuto, C., Yu, K., Jaggi, M., Musat, C., and Salzmann, M. Evaluating the search phase of neural architecture search. *arXiv preprint arXiv:1902.08142*, 2019.

Shaban, A., Cheng, C.-A., Hatch, N., and Boots, B. Truncated back-propagation for bilevel optimization. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1723–1732. PMLR, 2019.

Tan, M. and Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114, 2019.

Wang, R., Cheng, M., Chen, X., Tang, X., and Hsieh, C.-J. Rethinking architecture selection in differentiable nas. In *ICLR*, 2021a.

Wang, R., Cheng, M., Chen, X., Tang, X., and Hsieh, C.-J. Rethinking architecture selection in differentiable nas. In *International Conference on Learning Representations*, 2021b.

Xie, S., Zheng, H., Liu, C., and Lin, L. Snas: stochastic neural architecture search. *ICLR*, 2019.

Xu, Y., Xie, L., Zhang, X., Chen, X., Qi, G.-J., Tian, Q., and Xiong, H. Pc-darts: Partial channel connections for memory-efficient architecture search. In *ICLR*, 2020.

Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., and Hutter, F. Nas-bench-101: Towards reproducible neural architecture search. In *ICML*, pp. 7105–7114, 2019.

Zela, A., Elsken, T., Saikia, T., Marrakchi, Y., Brox, T., and Hutter, F. Understanding and robustifying differentiable architecture search. In *ICLR*, 2020a.

Zela, A., Siems, J., and Hutter, F. Nas-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search. In *ICLR*, 2020b.

Zhang, M., Li, H., Pan, S., Chang, X., Ge, Z., and Su, S. Differentiable neural architecture search in equivalent space with exploration enhancement. In *NeurIPS*, 2020a.

Zhang, M., Li, H., Pan, S., Chang, X., and Su, S. Overcoming multi-model forgetting in one-shot nas with diversity maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7809–7818, 2020b.

Zhang, M., Li, H., Pan, S., Chang, X., Zhou, C., Ge, Z., and Su, S. W. One-shot neural architecture search: Maximising diversity to overcome catastrophic forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020c.

Zhang, M., Li, H., Pan, S., Liu, T., and Su, S. One-shot neural architecture search via novelty driven sampling. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. International Joint Conferences on Artificial Intelligence Organization, 2020d.

Zheng, X., Ji, R., Tang, L., Zhang, B., Liu, J., and Tian, Q. Multinomial distribution learning for effective neural architecture search. In *International Conference on Computer Vision (ICCV)*, 2019.

Zhu, F., Liang, X., Zhu, Y., Chang, X., and Liang, X. SOON: scenario oriented object navigation with graph-based exploration. *CoRR*, abs/2103.17138, 2021.

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. In *ICLR*, 2017.

A. Proof

Proof of Lemma 2: Based on the implicit function theorem (Lorraine et al., 2020), or we simply set $\frac{\mathcal{L}_1(w^*,\alpha)}{\partial w} = 0$ since the model weights w achieved the local optimal in the training set with α, we have:

$$\frac{\partial \mathcal{L}_1(w^*(\alpha),\alpha)}{\partial w} = 0, \tag{10}$$

and we have

$$\begin{aligned}\frac{\partial}{\partial \alpha}\left(\frac{\partial \mathcal{L}_1(w^*(\alpha),\alpha)}{\partial w}\right) &= 0,\\ \frac{\partial^2 \mathcal{L}_1}{\partial \alpha \partial w} + \frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\frac{\partial (w^*(\alpha))}{\partial \alpha} &= 0,\\ \frac{\partial (w^*(\alpha))}{\partial \alpha} &= -\left[\frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^{-1}\frac{\partial^2 \mathcal{L}_1}{\partial \alpha \partial w}.\end{aligned} \tag{11}$$

In this way, the hypergradient could be formulated as

$$\nabla_\alpha \mathcal{L}_2 = \frac{\partial \mathcal{L}_2}{\partial \alpha} - \frac{\partial \mathcal{L}_2}{\partial w}\left[\frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^{-1}\frac{\partial^2 \mathcal{L}_1}{\partial \alpha \partial w}. \tag{12}$$

□

Proof of Corollary 1: The key in this proposition is to use the Neumann series to approximate the $\left[\frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^{-1}$.

Based on the Neumann series approximation, for $\|I - A\| < 1$, we have:

$$A^{-1} = \sum_{k=0}^{\infty}(I-A)^k. \tag{13}$$

Based Assumption 2.1, we have $\frac{\partial^2 \mathcal{L}_1}{\partial w \partial w} < L_1^{\nabla_w}$. With $\gamma < \frac{1}{L_1^{\nabla_w}}$, we have $\left\|I - \gamma\frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right\| < 1$ (Shaban et al., 2019; Lorraine et al., 2020). When we conduct the Neumann series approximation for $\left[\frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^{-1}$ in the optimal point, we have:

$$\left[\frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^{-1} = \gamma(I - I + \gamma\frac{\partial^2 \mathcal{L}_1}{\partial w \partial w})^{-1} = \gamma\sum_{j=0}^{\infty}\left[I - \gamma\frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^j. \tag{14}$$

So that:

$$\nabla_\alpha \mathcal{L}_2 = \frac{\partial \mathcal{L}_2}{\partial \alpha} - \gamma\frac{\partial \mathcal{L}_2}{\partial w}\sum_{j=0}^{\infty}\left[I - \gamma\frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^j\frac{\partial^2 \mathcal{L}_1}{\partial \alpha \partial w}. \tag{15}$$

□

Proof of Theorem 1 Based on the Eq. (8) and (7), we have

$$\nabla_\alpha \mathcal{L}_2 - \nabla_\alpha \tilde{\mathcal{L}}_2 = \gamma\frac{\partial \mathcal{L}_2}{\partial w}\sum_{j=K+1}^{\infty}\left[I - \gamma\frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^j\frac{\partial^2 \mathcal{L}_1}{\partial \alpha \partial w}. \tag{16}$$

Since the $\mathcal{L}_1$ is μ-strongly convex, and $\gamma\mu I \preceq \gamma\frac{\partial^2 \mathcal{L}_1}{\partial w \partial w} \preceq I$, we have

$$\sum_{j=K+1}^{\infty}\left[I - \gamma\frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^j \leq \sum_{j=K+1}^{\infty}\left[I - \gamma\mu\right]^j. \tag{17}$$

Based on the sum of geometric sequence, we have

$$\sum_{j=K+1}^{\infty}\left[I-\gamma\frac{\partial^2\mathcal{L}_1}{\partial w\partial w}\right]^j \leq \frac{1}{\gamma\mu}(1-\gamma\mu)^{K+1}. \tag{18}$$

Since $\frac{\partial\mathcal{L}_2}{\partial w}$ and $\frac{\partial^2\mathcal{L}_1}{\partial\alpha\partial w}$ are bounded, we have

$$\left\|\nabla_\alpha\mathcal{L}_2-\nabla_\alpha\tilde{\mathcal{L}}_2\right\| \leqslant C_{\mathcal{L}_1^{w\alpha}}\, C_{\mathcal{L}_2^{w}}\, \frac{1}{\mu}(1-\gamma\mu)^{K+1}. \tag{19}$$

□

Proof: Corollary 2 Based on the definitions, the hypergradient of truncated back-propagation and the proposed Neumann approximation based hypergradient are defined in Eq.(4) and Eq.(8). When we assume that w_t has converged to a stationary point w^* in the last K steps, we have

$$\begin{aligned}
w_i(\alpha)=w_j(\alpha)&=w^*(\alpha), && \textit{for all } i,j\in[T-K+1,T];\\
\frac{\partial\Phi(w_i,\alpha)}{\partial w_i}=\frac{\partial\Phi(w_j,\alpha)}{\partial w_j}&=\frac{\partial\Phi(w^*(\alpha),\alpha)}{\partial w^*(\alpha)}=A_T, && \textit{for all } i,j\in[T-K+1,T];\\
\frac{\partial\Phi(w_i,\alpha)}{\partial \alpha}=\frac{\partial\Phi(w_j,\alpha)}{\partial \alpha}&=\frac{\partial\Phi(w^*(\alpha),\alpha)}{\partial \alpha}=B_T, && \textit{for all } i,j\in[T-K+1,T].
\end{aligned} \tag{20}$$

Now the truncated back-propagation could be formulated as:

$$\begin{aligned}
h_{T-K}&=\frac{\partial\mathcal{L}_2}{\partial\alpha}+\frac{\partial\mathcal{L}_2}{\partial w_T}\Big(\sum_{t=T-K+1}^{T}B_tA_{t+1}...A_T\Big)\\
&=\frac{\partial\mathcal{L}_2}{\partial\alpha}+\frac{\partial\mathcal{L}_2}{\partial w_T}\Big(\sum_{t=0}^{K}B_TA_T^t\Big).
\end{aligned} \tag{21}$$

We have

$$\begin{aligned}
A_T&=\frac{\partial\Phi(w^*(\alpha),\alpha)}{\partial w^*(\alpha)}=\frac{\partial(w^*-\eta\frac{\partial\mathcal{L}_1}{\partial w})}{\partial w^*}=I-\gamma\frac{\partial^2\mathcal{L}_1(w^*)}{\partial w\partial w},\\
B_T&=\frac{\partial\Phi(w^*(\alpha),\alpha)}{\partial \alpha}=\frac{\partial(w^*-\eta\frac{\partial\mathcal{L}_1}{\partial w})}{\partial \alpha}=-\gamma\frac{\partial^2\mathcal{L}_1(w^*)}{\partial \alpha\partial w}.
\end{aligned} \tag{22}$$

From the above, we have

$$\begin{aligned}
h_{T-K}&=\frac{\partial\mathcal{L}_2}{\partial\alpha}+\frac{\partial\mathcal{L}_2}{\partial w_T}\Big(\sum_{t=0}^{K}B_TA_T^t\Big)\\
&=\frac{\partial\mathcal{L}_2}{\partial\alpha}-\gamma\frac{\partial\mathcal{L}_2}{\partial w}\sum_{j=0}^{K}\left[I-\gamma\frac{\partial^2\mathcal{L}_1}{\partial w\partial w}\right]^j\frac{\partial^2\mathcal{L}_1}{\partial\alpha\partial w}\\
&=\nabla_\alpha\tilde{\mathcal{L}}_2.
\end{aligned} \tag{23}$$

□

Proof of Lemma 4: First, for $\forall(\alpha,\alpha')$, we have

$$\begin{aligned}
&\|\nabla_\alpha\mathcal{L}_2(w,\alpha)-\nabla_\alpha\mathcal{L}_2(w,\alpha')\|=\|\nabla_\alpha\mathcal{L}_2(\cdot,\alpha)-\nabla_\alpha\mathcal{L}_2(\cdot,\alpha')+\nabla_\alpha\mathcal{L}_2(w(\alpha),\cdot)-\nabla_\alpha\mathcal{L}_2(w(\alpha'),\cdot)\|\\
=&\|\nabla_\alpha\mathcal{L}_2(\cdot,\alpha)-\nabla_\alpha\mathcal{L}_2(\cdot,\alpha')+\nabla_w\mathcal{L}_2(w(\alpha),\cdot)\nabla_\alpha w(\alpha)-\nabla_w\mathcal{L}_2(w(\alpha'),\cdot)\nabla_\alpha w(\alpha')\|\\
\leq&\|\nabla_\alpha\mathcal{L}_2(\cdot,\alpha)-\nabla_\alpha\mathcal{L}_2(\cdot,\alpha')\|+\|\nabla_w\mathcal{L}_2(w(\alpha),\cdot)\nabla_\alpha w(\alpha)-\nabla_w\mathcal{L}_2(w(\alpha'),\cdot)\nabla_\alpha w(\alpha')\|.
\end{aligned} \tag{24}$$

Then we divide Eq.(24) to two parts. For the first part, based on the Assumption 1.2, we have:

$$\|\nabla_\alpha \mathcal{L}_2(\cdot,\alpha) - \nabla_\alpha \mathcal{L}_2(\cdot,\alpha')\| \leq L_2^{\nabla_\alpha}(\alpha-\alpha'). \tag{25}$$

And for the second part of Eq.(24), we have

$$\begin{aligned}&\|\nabla_w \mathcal{L}_2(w(\alpha),\cdot)\nabla_\alpha w(\alpha) - \nabla_w \mathcal{L}_2(w(\alpha'),\cdot)\nabla_\alpha w(\alpha')\|\\ =&\|\nabla_w \mathcal{L}_2(w(\alpha),\cdot)\nabla_\alpha w(\alpha) - \nabla_w \mathcal{L}_2(w(\alpha'),\cdot)\nabla_\alpha w(\alpha) - \nabla_w \mathcal{L}_2(w(\alpha'),\cdot)\nabla_\alpha w(\alpha') + \nabla_w \mathcal{L}_2(w(\alpha'),\cdot)\nabla_\alpha w(\alpha)\|\\ \leq&\|\nabla_w \mathcal{L}_2(w(\alpha'),\cdot) - \nabla_w \mathcal{L}_2(w(\alpha'),\cdot)\|\,\|\nabla_\alpha w(\alpha)\| + \|\nabla_w \mathcal{L}_2(w(\alpha'),\cdot)\|\,\|\nabla_\alpha w(\alpha) - \nabla_\alpha w(\alpha')\|.\end{aligned} \tag{26}$$

Based Assumption 1.3, we have

$$\|\nabla_w \mathcal{L}_2(w(\alpha'),\cdot) - \nabla_w \mathcal{L}_2(w(\alpha'),\cdot)\| \leq L_2^{\nabla_w}\|w(\alpha)-w(\alpha')\|, \tag{27}$$

and based Assumption 2.2 that we have

$$\|w(\alpha)-w(\alpha')\| \leq L_w\|\alpha-\alpha'\|, \quad \text{and} \quad \|\nabla_\alpha w(\alpha) - \nabla_\alpha w(\alpha')\| \leq L_{\nabla_\alpha w}\|\alpha-\alpha'\|. \tag{28}$$

Based on Assumption 1.3, we know $\nabla_w \mathcal{L}_2(w(\alpha'),\cdot)$ is bounded that $\nabla_w \mathcal{L}_2(w(\alpha'),\cdot) \leq L_2^w$. $\nabla_\alpha w(\alpha)$ is also bounded by $\|\nabla_\alpha w(\alpha)\| \leq L_w$. In this way, Eq.(26) could be rephrased as:

$$\|\nabla_w \mathcal{L}_2(w(\alpha),\cdot)\nabla_\alpha w(\alpha) - \nabla_w \mathcal{L}_2(w(\alpha'),\cdot)\nabla_\alpha w(\alpha')\| \leq L_2^{\nabla_w}L_w^2\|\alpha-\alpha'\| + L_2^w L_{\nabla_\alpha w}\|\alpha-\alpha'\|. \tag{29}$$

Based on Eq. (24), Eq. (25) and (29) we have

$$\|\nabla_\alpha \mathcal{L}_2(w,\alpha) - \nabla_\alpha \mathcal{L}_2(w,\alpha')\| \leq (L_2^{\nabla_\alpha} + L_2^{\nabla_w}L_w^2 + L_2^w L_{\nabla_\alpha w})\|\alpha-\alpha'\|. \tag{30}$$

Therefore, Lemma 4 is proved.

□

Proof of Theorem 2: We first define the noise term between the stochastic estimate $\nabla_\alpha \mathcal{L}_2^i$ and the true gradient $\nabla_\alpha \mathcal{L}_2$ as:

$$\varepsilon_i = \nabla_\alpha \mathcal{L}_2 - \nabla_\alpha \mathcal{L}_2^i, \tag{31}$$

and the error between the approximated hypergradient $\nabla_\alpha \tilde{\mathcal{L}}_2$ and the exact hypergradient $\nabla_\alpha \mathcal{L}_2$ as:

$$e_m = \nabla_\alpha \mathcal{L}_2(w^*(\alpha_m),\alpha_m) - \nabla_\alpha \tilde{\mathcal{L}}_2(w^*(\alpha_m),\alpha_m). \tag{32}$$

We then prove that $\nabla_\alpha \mathcal{L}_2^i(w^*(\alpha_m),\alpha_m)$ is an unbiased estimate of $\nabla_\alpha \mathcal{L}_2(w^*(\alpha_m),\alpha_m)$ that:

$$E[\nabla_\alpha \mathcal{L}_2^i(w^*(\alpha_m),\alpha_m) \mid \alpha_m] = \nabla_\alpha \mathcal{L}_2(w^*(\alpha_m),\alpha_m). \tag{33}$$

Based on IFT in Eq.(7), we have

$$\nabla_\alpha \mathcal{L}_2^i(w^*(\alpha_m),\alpha_m) = \frac{\partial \mathcal{L}_2^i(w^*(\alpha_m),\alpha_m)}{\partial \alpha} - \frac{\partial \mathcal{L}_2^i(w^*(\alpha_m),\alpha_m)}{\partial w}\left[\frac{\partial^2 \mathcal{L}_1^j(w^*(\alpha_m),\alpha_m)}{\partial w \partial w}\right]^{-1}\frac{\partial^2 \mathcal{L}_1^j(w^*(\alpha_m),\alpha_m)}{\partial \alpha \partial w}. \tag{34}$$

So that

$$\begin{aligned}&E\left[\nabla_\alpha \mathcal{L}_2^i(w^*(\alpha_m),\alpha_m) \mid \alpha_m\right]\\ =&E\left[\frac{\partial \mathcal{L}_2^i(w^*(\alpha_m),\alpha_m)}{\partial \alpha} - \frac{\partial \mathcal{L}_2^i(w^*(\alpha_m),\alpha_m)}{\partial w}\left[\frac{\partial^2 \mathcal{L}_1^j(w^*(\alpha_m),\alpha_m)}{\partial w \partial w}\right]^{-1}\frac{\partial^2 \mathcal{L}_1^j(w^*(\alpha_m),\alpha_m)}{\partial \alpha \partial w} \mid \alpha_m\right].\end{aligned} \tag{35}$$

Based on the linear assumption for $\mathcal{L}_1^j$ in the condition 4 of the Theorem 2, we have $\frac{\partial^2 \mathcal{L}_1^j(w^*(\alpha_m),\alpha_m)}{\partial w \partial w} = \frac{\partial^2 \mathcal{L}_1(w^*(\alpha_m),\alpha_m)}{\partial w \partial w}$, and

$$
\begin{aligned}
&E\left[\nabla_\alpha \mathcal{L}_2^i(w^*(\alpha_m),\alpha_m) \mid \alpha_m\right] \\
&= \frac{1}{R}\sum_{i=1}^{R} \frac{\partial \mathcal{L}_2^i(w^*(\alpha_m),\alpha_m)}{\partial \alpha} - \frac{1}{R}\sum_{i=1}^{R} \frac{\partial \mathcal{L}_2^i(w^*(\alpha_m),\alpha_m)}{\partial w}\left[\frac{\partial^2 \mathcal{L}_1(w^*(\alpha_m),\alpha_m)}{\partial w \partial w}\right]^{-1} \frac{1}{J}\sum_{j=1}^{J} \frac{\partial^2 \mathcal{L}_1^j(w^*(\alpha_m),\alpha_m)}{\partial \alpha \partial w} \\
&= \frac{\partial \mathcal{L}_2(w^*(\alpha_m),\alpha_m)}{\partial \alpha} - \frac{\partial \mathcal{L}_2(w^*(\alpha_m),\alpha_m)}{\partial w}\left[\frac{\partial^2 \mathcal{L}_1(w^*(\alpha_m),\alpha_m)}{\partial w \partial w}\right]^{-1} \frac{\partial^2 \mathcal{L}_1(w^*(\alpha_m),\alpha_m)}{\partial \alpha \partial w} \\
&= \nabla_\alpha \mathcal{L}_2(w^*(\alpha_m),\alpha_m).
\end{aligned} \tag{36}
$$

Based on the Lemma 4, we know that $\nabla_\alpha \mathcal{L}_2(w^*(\alpha_m),\alpha_m)$ is Lipschitz continuous with $L_{\nabla_\alpha \mathcal{L}_2} = L_2^{\nabla_\alpha} + L_2^{\nabla_w} L_w^2 + L_2^w L_{\nabla_\alpha w}$. Based on Lipschitz condition, we have

$$
\begin{aligned}
&E\left[\mathcal{L}_2(w^*(\alpha_{m+1}),\alpha_{m+1}) \mid \alpha_m\right] \leq E\left[\mathcal{L}_2(w^*(\alpha_m),\alpha_m) \mid \alpha_m\right] \\
&+ E\left[\langle \nabla_\alpha \mathcal{L}_2(w^*(\alpha_m),\alpha_m), \alpha_{m+1}-\alpha_m \rangle \mid \alpha_m\right] + \frac{L_{\nabla_\alpha \mathcal{L}_2}}{2} E\left[\|\alpha_{m+1}-\alpha_m\|^2\right] \\
&= \mathcal{L}_2(w^*(\alpha_m),\alpha_m) + \left\langle E\left[\nabla_\alpha \mathcal{L}_2(w^*(\alpha_m),\alpha_m)\right], -\gamma_{\alpha_m} E\left[\nabla_\alpha \mathcal{L}_2^{i'}(w^*(\alpha_m),\alpha_m) \mid \alpha_m\right]\right\rangle \\
&+ \frac{L_{\nabla_\alpha \mathcal{L}_2}}{2}\gamma_{\alpha_m}^2 E\left[\left\|\nabla_\alpha \mathcal{L}_2^{i'}(w^*(\alpha_m),\alpha_m)\right\|^2\right].
\end{aligned} \tag{37}
$$

From our definitions, we have

$$
\begin{aligned}
E\left[\nabla_\alpha \mathcal{L}_2(w^*(\alpha_m),\alpha_m)\right] &= E\left[\nabla_\alpha \tilde{\mathcal{L}}_2(w^*(\alpha_m),\alpha_m) + e_m\right] = E\left[\nabla_\alpha \tilde{\mathcal{L}}_2(w^*(\alpha_m),\alpha_m)\right] + E\left[e_m\right], \\
E\left[\nabla_\alpha \hat{\mathcal{L}}_2^i(w^*(\alpha_m),\alpha_m) \mid \alpha_m\right] &= E\left[\nabla_\alpha \tilde{\mathcal{L}}_2(w^*(\alpha_m),\alpha_m) - \varepsilon_m \mid \alpha_m\right] = E\left[\nabla_\alpha \tilde{\mathcal{L}}_2(w^*(\alpha_m),\alpha_m)\right], \\
E\left[\left\|\nabla_\alpha \hat{\mathcal{L}}_2^i(w^j(\alpha_m),\alpha_m)\right\|^2 \mid \alpha_m\right] &= E\left[\left\|\nabla_\alpha \tilde{\mathcal{L}}_2(w^*(\alpha_m),\alpha_m) - \varepsilon_m\right\|^2\right] = E\left[\left\|\nabla_\alpha \tilde{\mathcal{L}}_2(w^*(\alpha_m),\alpha_m)\right\|^2\right] + E\left[\|\varepsilon_m\|^2\right],
\end{aligned} \tag{38}
$$

since $E(\varepsilon_m) = 0$. In this way, we have

$$
\begin{aligned}
E\left[\mathcal{L}_2(w^*(\alpha_{m+1}),\alpha_{m+1}) \mid \alpha_m\right] \leq E\left[\mathcal{L}_2(w^*(\alpha_m),\alpha_m) \mid \alpha_m\right] &- \gamma_{\alpha_m} E\left[\left\|\nabla_\alpha \tilde{\mathcal{L}}_2(w^*(\alpha_m),\alpha_m)\right\|^2\right] \\
&- \gamma_{\alpha_m} E\left\langle e_m, \nabla_\alpha \tilde{\mathcal{L}}_2(w^*(\alpha_m),\alpha_m)\right\rangle + \frac{L_{\nabla_\alpha \mathcal{L}_2}}{2}\gamma_{\alpha_m}^2 E\left[\left\|\nabla_\alpha \tilde{\mathcal{L}}_2(w^*(\alpha_m),\alpha_m)\right\|^2\right] \\
&+ \frac{L_{\nabla_\alpha \mathcal{L}_2}}{2}\gamma_{\alpha_m}^2 E\left[\|\varepsilon_m\|^2\right].
\end{aligned} \tag{39}
$$

Based on Theorem 1, we have $\|e_m\| \leqslant C_{\mathcal{L}_1^{w\alpha}} C_{\mathcal{L}_2^w} \frac{1}{\mu}(1-\gamma\mu)^{K+1}$. In this way, for all $\nabla_\alpha \tilde{\mathcal{L}}_2(w^*(\alpha_m),\alpha_m)$, we have

$$
\begin{aligned}
\left\langle e_m, \nabla_\alpha \tilde{\mathcal{L}}_2(w^*(\alpha_m),\alpha_m)\right\rangle &\geq -C_{\mathcal{L}_1^{w\alpha}} C_{\mathcal{L}_2^w} \frac{1}{\mu}(1-\gamma\mu)^{K+1}\left\|\nabla_\alpha \tilde{\mathcal{L}}_2\right\| \\
&= -\frac{C_{\mathcal{L}_1^{w\alpha}} C_{\mathcal{L}_2^w} (1-\gamma\mu)^{K+1}}{\mu\left\|\nabla_\alpha \tilde{\mathcal{L}}_2\right\|}\left\|\nabla_\alpha \tilde{\mathcal{L}}_2\right\|^2 \\
&= -P\left\|\nabla_\alpha \tilde{\mathcal{L}}_2\right\|^2,
\end{aligned} \tag{40}
$$

where $P = \frac{C_{\mathcal{L}_1^{w\alpha}} C_{\mathcal{L}_2^{w}} (1-\gamma\mu)^{K+1}}{\mu \left\| \nabla_\alpha \tilde{\mathcal{L}}_2 \right\|}$. In this way, we have:

$$\begin{aligned} E\left[\mathcal{L}_2(w^*(\alpha_{m+1}), \alpha_{m+1})\right] &\leq E\left[\mathcal{L}_2(w^*(\alpha_m), \alpha_m)\right] - \gamma_{\alpha_m}(1-P)E\left[\left\|\nabla_\alpha \tilde{\mathcal{L}}_2\right\|^2\right] \\ &+ \frac{L_{\nabla_\alpha \mathcal{L}_2}}{2}\gamma_{\alpha_m}^2(1+D)E\left[\left\|\nabla_\alpha \tilde{\mathcal{L}}_2\right\|^2\right] \\ &\leq E\left[\mathcal{L}_2(w^*(\alpha_m), \alpha_m)\right] - \gamma_{\alpha_m}[(1-P) - \frac{L_{\nabla_\alpha \mathcal{L}_2}}{2}\gamma_{\alpha_m}(1+D)]E\left[\left\|\nabla_\alpha \tilde{\mathcal{L}}_2\right\|^2\right]. \end{aligned} \tag{41}$$

If we choose γ_{α_m} to make $(1-P) - \frac{L_{\nabla_\alpha \mathcal{L}_2}}{2}\gamma_{\alpha_m}(1+D) > 0$, we have $\gamma_{\alpha_m} < \frac{(1-P)}{\frac{L_{\nabla_\alpha \mathcal{L}_2}}{2}(1+D)}$. In addition, since the learning rate should be positive, we should make that $1-P>0$, which could be reached by choose appropriate γ and K that $\frac{C_{\mathcal{L}_1^{w\alpha}} C_{\mathcal{L}_2^{w}} (1-\gamma\mu)^{K+1}}{\mu \left\| \nabla_\alpha \tilde{\mathcal{L}}_2 \right\|} < 1$, where $0 < 1-\gamma\mu \leq 1$. In this way, we could find that $\mathcal{L}_2$ is decreasing with α_m, and we know that with sufficiently large m, $\mathcal{L}_2$ will decrease and converge since $\mathcal{L}_2$ is bounded.

Furthermore, we have:

$$\begin{aligned} &E\left[\mathcal{L}_2(w^*(\alpha_m), \alpha_m)\right] - E\left[\mathcal{L}_2(w^*(\alpha_{m+1}), \alpha_{m+1})\right] \\ \geq &\gamma_{\alpha_m}[(1-P) - \frac{L_{\nabla_\alpha \mathcal{L}_2}}{2}\gamma_{\alpha_m}(1+D)]E\left[\left\|\nabla_\alpha \tilde{\mathcal{L}}_2(w^*(\alpha_m), \alpha_m)\right\|^2\right]. \end{aligned} \tag{42}$$

By telescoping sum, we can show that

$$E\left[\mathcal{L}_2(w^*(\alpha_0), \alpha_0)\right] - E\left[\mathcal{L}_2(w^*(\alpha_m), \alpha_m)\right] \geq \sum_{k=0}^{K} q_t E\left[\left\|\nabla_\alpha \tilde{\mathcal{L}}_2(w^*(\alpha_m), \alpha_m)\right\|^2\right], \tag{43}$$

where $q_t = \gamma_{\alpha_m}[(1-P) - \frac{L_{\nabla_\alpha \mathcal{L}_2}}{2}\gamma_{\alpha_m}(1+D)] > 0$. Since $\mathcal{L}_2$ is bounded, we have $\sum_{\substack{k=0 \\ K\to\infty}}^{K} q_t E\left[\left\|\nabla_\alpha \tilde{\mathcal{L}}_2(w^*(\alpha_m), \alpha_m)\right\|^2\right] < \infty$. In addition, based on condition 3, we have $\sum_{\substack{k=0 \\ K\to\infty}}^{K} q_t = \infty$, which imply that $\lim_{k\to\infty} E\left[\left\|\nabla_\alpha \tilde{\mathcal{L}}_2(w^*(\alpha_m), \alpha_m)\right\|\right] = 0$, so as $\lim_{m\to\infty} E\left[\left\|\nabla_\alpha \hat{\mathcal{L}}_2^i(w^j(\alpha_m), \alpha_m)\right\|\right] = 0$.

□

B. Practical implementation of hypergradient

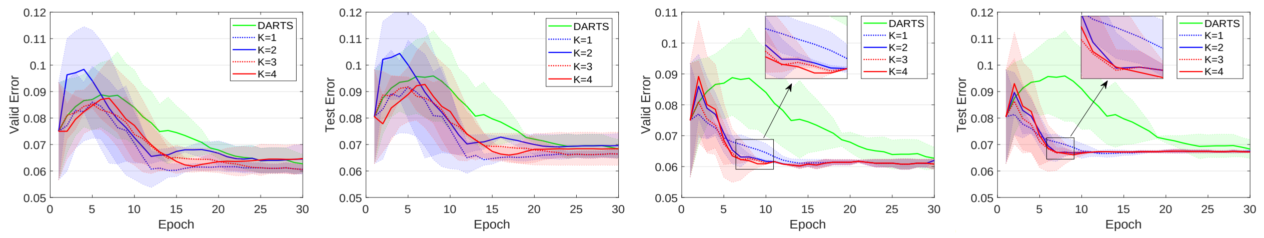
As described, our iDARTS is built based on the DARTS framework with reformulation the hypergradient calculation as:

$$\nabla_\alpha \tilde{\mathcal{L}}_2 = \frac{\partial \mathcal{L}_2}{\partial \alpha} - \gamma \frac{\partial \mathcal{L}_2}{\partial w} \sum_{k=0}^{K} \left[I - \gamma \frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^k \frac{\partial^2 \mathcal{L}_1}{\partial \alpha \partial w}. \tag{44}$$

where the different part is the $\left[I - \gamma \frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^k$. As known, it is costly to calculate the Hessian matrix $\frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}$ and $\frac{\partial^2 \mathcal{L}_1}{\partial \alpha \partial w}$ for a large neural network, and we propose two approximations to reduce the computational cost for pracyical implementation, as described below.

Approximation 1: Although it is hard to directly calculate the Hessian matrix $\frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}$, we could consider Hessian-vector product technique with autograd to calculate $\frac{\partial \mathcal{L}_2}{\partial w} \cdot \frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}$. In this way, we can calculate $\frac{\partial \mathcal{L}_2}{\partial w} \sum_{k=0}^{K} \left[I - \gamma \frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^k$ step by step:

$$\frac{\partial \mathcal{L}_2}{\partial w} \sum_{k=0}^{K} \left[I - \gamma \frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^k = \sum_{k=0}^{K} \frac{\partial \mathcal{L}_2}{\partial w} \left[I - \gamma \frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^k = \sum_{k=0}^{K} V_0 \left[I - \gamma H\right]^k = V_0 + V_1 + V_2 + ... + V_k. \tag{45}$$



(a) Validation error with T=1 (b) Test errors with T=1 (c) Validation error with T=5 (d) Test errors with T=5

Figure 4. Ablation study on K for iDARTS with $T = 1$ and $T = 5$ on NAS-Bench-1Shot1.

where we define $V_0 = \frac{\partial \mathcal{L}_2}{\partial w}$, $H = \gamma \frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}$, and $V_1 = V_0(I - H), V_2 = V_1(I - H), ..., V_K = V_{K-1}(I - H)$. We can find that, $\frac{\partial \mathcal{L}_2}{\partial w} \sum_{k=0}^{K} \left[I - \gamma \frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^k$ could be calculated with K steps of Hessian-vector product.

Approximation 2: Apart from the Hessian matrix $\frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}$, it is also costly to calculate $\frac{\partial^2 \mathcal{L}_1}{\partial \alpha \partial w}$ for large neural networks, and we follow DARTS to use the Taylor expansion to approximate $\frac{\partial^2 \mathcal{L}_1}{\partial \alpha \partial w}$. After calculating $\frac{\partial \mathcal{L}_2}{\partial w} \sum_{k=0}^{K} \left[I - \gamma \frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^k$, considering the function $\frac{\partial \mathcal{L}_1(w,\alpha)}{\partial \alpha}$ with Taylor expansion, we have

$$\begin{aligned} \frac{\partial \mathcal{L}_1(w + \epsilon A, \alpha)}{\partial \alpha} &= \frac{\partial \mathcal{L}_1(w, \alpha)}{\partial \alpha} + \frac{\partial^2 \mathcal{L}_1(, \alpha)}{\partial \alpha \partial w} \epsilon A + ..., \\ \frac{\partial \mathcal{L}_1(w - \epsilon A, \alpha)}{\partial \alpha} &= \frac{\partial \mathcal{L}_1(w, \alpha)}{\partial \alpha} - \frac{\partial^2 \mathcal{L}_1(w, \alpha)}{\partial \alpha \partial w} \epsilon A + ..., \end{aligned} \tag{46}$$

where ϵ is a very small scalar. When we replace A with $\frac{\partial \mathcal{L}_2}{\partial w} \sum_{k=0}^{K} \left[I - \gamma \frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^k$, we have

$$\frac{\partial \mathcal{L}_2}{\partial w} \sum_{k=0}^{K} \left[I - \gamma \frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^k \frac{\partial^2 \mathcal{L}_1}{\partial \alpha \partial w} = \frac{\frac{\partial \mathcal{L}_1(w+\epsilon A,\alpha)}{\partial \alpha} - \frac{\partial \mathcal{L}_1(w-\epsilon A,\alpha)}{\partial \alpha}}{2\epsilon}. \tag{47}$$

As described, the proposed approximated hypergradient $\nabla_\alpha \tilde{\mathcal{L}}_2$ is easy to implement based on the DARTS framework with only replacing $\frac{\partial \mathcal{L}_2}{\partial w}$ to $\frac{\partial \mathcal{L}_2}{\partial w} \sum_{k=0}^{K} \left[I - \gamma \frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^k$, which could be computed using the the Hessian-vector product technique.

Therefor, we can practically implement our approximated hypergradient $\nabla_\alpha \tilde{\mathcal{L}}_2$, so as the stochastic approximated hypergradient $\nabla_\alpha \hat{\mathcal{L}}_2^i(w^j(\alpha), \alpha)$ with minibatches based on the DARTS framework [2].

C. Ablation study on the number of approximation terms K

As we described before, there are two additional hyperparameters in our practical iDARTS, the inner optimization steps T and the number of terms for the approximation in Eq.(8). We have analyzed T in previous experiments. In this section, we analyze another hyperparameter K on the NAS-Bench-1Shot1 benchmark dataset. In the first experiment, we set a default hyperparameter $T = 1$ the same as DARTS for the inner supernet training to remove the bias from T. From Eq.(8) and (3), we could further find that the hypergradient calculation in our iDARTS with $T = 1$ and $K = 0$ is the same as DARTS. Figure 4 (a) (b) plots the performance of iDARTS with different K on the NAS-Bench-1Shot1. As shown, our iDARTS is very robust to K with limited training steps $T = 1$, where iDARTS with different K all outperform the DARTS baseline with the same inner training steps $T = 1$, showing the superiority of the proposed approximation over DARTS. Another interesting finding is that, our iDARTS with $K = 1$ and $T = 1$ even achieve slightly more competitive results than $K > 1$. An underlying reason is that, when the inner training step is too small, it is hard to achieve the local optimal w^* and the corresponding hypergradient is not accurate.

[2]The codes and training log files could be found in the supplementary material. The best trained models on CIFAR-10, CIFR-100, and ImageNet could be found https://github.com/MiaoZhang0525/iDARTS.

Table 3. Ablation study on K for iDARTS with on NAS-Bench-201.

Method	CIFAR-10		CIFAR-100		ImageNet-16-120	
	Valid(%)	Test(%)	Valid(%)	Test(%)	Valid(%)	Test(%)
DARTS($T=1$, $K=0$)	39.77±0.00	54.30±0.00	15.03±0.00	15.61±0.00	16.43±0.00	16.32±0.00
iDARTS($T=1$, $K=1$)	86.85±0.93	89.67±1.31	64.09±2.92	64.17±3.26	36.26±5.71	36.11± 5.77
iDARTS($T=4$, $K=0$)	87.31±1.33	90.36±1.79	64.76±2.54	64.43±2.47	32.53±1.31	32.42±1.54
iDARTS($T=4$, $K=1$)	89.30±1.47	92.44±1.14	67.88±1.86	68.17±2.81	37.11±7.79	36.61±7.47
iDARTS($T=4$, $K=2$)	89.86±0.60	93.58±0.32	70.57±0.24	70.83±0.48	40.38±0.59	40.89±0.68
iDARTS($T=4$, $K=3$)	89.35±0.03	92.29±0.26	68.51±0.77	68.58±1.18	42.37±0.48	42.26±0.41

To further investigate the effectiveness of the proposed approximation, we consider setting enough inner training steps with $T=5$, and Figure 4 (c) (d) plots the performance of iDARTS with different K on the NAS-Bench-1Shot1 under $T=5$. The first impression from Figure 4 is that increasing inner training steps could significantly improve the performance, where all cases with $T=5$ generally outperform $T=1$. Another interesting finding is that, with enough inner training steps, the number of approximation terms K has a positive impact on the performance of iDARTS. As shown in Figure 4 (c) (d), increasing K also helps iDARTS converge to excellent solutions faster, verifying that the proposed $\nabla_\alpha \hat{\mathcal{L}}_2^i$ could asymptotically approach to the exact hypergradient $\nabla_\alpha \mathcal{L}_2^i$ with the increase of approximation term K. Besides, we can find that, $K=2$ is large enough to result in competitive performance for our iDARTS on NAS-Bench-1shot1, which results in similar performance as $K \geq 3$.

We also conduct an ablation study on NAS-Bench-201 dataset to analyse the hyperparameter K, and Table 3 summarizes the performance of iDARTS on NAS-Bench-201 with a different number of approximation term K. The results in Table 3 are similar to those on the NAS-Bench-1Shot1 dataset, also showing that K has a positive impact on the performance of iDARTS. Firstly, we can find that, with the same inner training steps $T=1$ as DARTS baseline, our iDARTS ($T=1$, $K=1$) with one approximation term outperform DARTS by large margins in this case, verifying the superiority of the proposed approximation over DARTS. Secondly, the results in Table 3 also demonstrate that considering more approximation terms does indeed help improve our iDARTS to a certain degree. With enough inner training steps, the performance of iDARTS increases with K from 0 to 2. Another interesting finding is that the performance of iDARTS does not always increase with the K, and there is a decrease for $K \geq 3$. One underlying reason may be that, the iDARTS with smaller K brings more noises into the hypergradient, which in turn enhances the exploration. Several recent works (Chen & Hsieh, 2020; Zhang et al., 2020a) show the importance of the exploration in the differentiable NAS, where adding more noises into the hypergradient could improve the performance. Our experimental results suggest that a $K=2$ achieves an excellent trade-off between the accuracy of hypergradient and the exploration, thus achieving the competitive performance on the NAS-Bench-201 dataset.

D. Experimental settings in all experiments

In the first experimental set, we choose the third search space of NAS-Bench-1Shot1 (Zela et al., 2020b) to analyze iDARTS, since it is much more complicated than the remaining two search spaces and is a better case to identify the advantages of iDARTS. In Section 5.1, we analyzed the hyperparameter T for our iDARTS and compared it with baseline on the NAS-Bench-1Shot1, and we set another hyperparameter $K=3$ in all cases. In Appendix C, we further conduct the ablation study to investigate another important hyperparameter K, where we consider two cases with $T=1$ and $T=5$, and the remaining experimental settings are the same as the default settings.

In the second experimental set, we choose the NAS-Bench-201 dataset (Dong & Yang, 2020) to analyze differentiable NAS methods. In Section 5.2, we first conduct a comparison experiment with several NAS baselines, and the hyperparameters for our iDARTS in this experiments are $T=4$, $K=2$, and γ=0.01. Then we conduct a series ablation studies to investigate three important hyperparameters, inner supernet training steps T, supernet learning rate γ, and architecture learning rate γ_α. In the experiment for the investigation of T, see Figure 2 (a), we set $K=1$ and other hyperparameters are default settings. In the Figure 2 (b) and (c), we set $T=4$ and $K=1$ to investigate both the supernet learning rate γ and architecture learning rate γ_α. In Appendix C, we also analyze the impact of K in iDARTS on NAS-Bench-201 dataset, where we set $T=4$ and γ=0.01, and the remaining settings are the default.

In the common DARTS search space, we follow the experimental settings in (Liu et al., 2019) to compare with the state-of-the-art NAS methods. We search for micro-cell structures on CIFAR-10 to stack more cells to form the final structure for

Table 4. An overview of different hypergradient approximations.

Method	Steps	Memory Cost	Hypergradient Calculation
Exact IFT hypergradient	∞	$\mathcal{O}(P+H)$	$\frac{\partial \mathcal{L}_2}{\partial \alpha} - \frac{\partial \mathcal{L}_2}{\partial w} \left[\frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^{-1} \frac{\partial^2 \mathcal{L}_1}{\partial \alpha \partial w}$
DARTS (Liu et al., 2019)	1	$\mathcal{O}(P+H)$	$\frac{\partial \mathcal{L}_2}{\partial \alpha} - \gamma \frac{\partial \mathcal{L}_2}{\partial w} \frac{\partial^2 \mathcal{L}_1}{\partial \alpha \partial w}$
$T_1 - T_2$ (Luketina et al., 2016)	1	$\mathcal{O}(P+H)$	$\frac{\partial \mathcal{L}_2}{\partial \alpha} - \frac{\partial \mathcal{L}_2}{\partial w} [I]^{-1} \frac{\partial^2 \mathcal{L}_1}{\partial \alpha \partial w}$
Reverse-mode (Franceschi et al., 2017)	T	$\mathcal{O}((P+H)T)$	$\frac{\partial \mathcal{L}_2}{\partial \alpha} + \frac{\partial \mathcal{L}_2}{\partial w_T} (\sum_{t=0}^{T} B_t A_{t+1} ... A_T)$
Truncated Reverse-mode (Shaban et al., 2019)	K	$\mathcal{O}((P+H)K)$	$\frac{\partial \mathcal{L}_2}{\partial \alpha} + \frac{\partial \mathcal{L}_2}{\partial w_T} (\sum_{t=T-K}^{T} B_t A_{t+1} ... A_T)$
Neumann Series (Bengio, 2000)	∞	$\mathcal{O}(P+H)$	$\frac{\partial \mathcal{L}_2}{\partial \alpha} - \gamma \frac{\partial \mathcal{L}_2}{\partial w} \sum_{j=0}^{\infty} \left[I - \gamma \frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^j \frac{\partial^2 \mathcal{L}_1}{\partial \alpha \partial w}$
Conjugate Gradient (Rajeswaran et al., 2019)	S	$\mathcal{O}(P+H)$	$\frac{\partial \mathcal{L}_2}{\partial \alpha} - \left(\text{argmin}_{\text{x}} \left\Vert \text{x} \frac{\partial^2 \mathcal{L}_1}{\partial w \partial w} - \frac{\partial \mathcal{L}_2}{\partial w} \right\Vert\right) \frac{\partial^2 \mathcal{L}_1}{\partial \alpha \partial w}$
Our Neumann approximation $\nabla_\alpha \hat{\mathcal{L}}_2^i$	K	$\mathcal{O}(P+H)$	$\frac{\partial \mathcal{L}_2}{\partial \alpha} - \gamma \frac{\partial \mathcal{L}_2}{\partial w} \sum_{j=0}^{K} \left[I - \gamma \frac{\partial^2 \mathcal{L}_1}{\partial w \partial w}\right]^j \frac{\partial^2 \mathcal{L}_1}{\partial \alpha \partial w}$

architecture evaluation. There are two types of cells with the unified search space: a normal cell α_{normal} and a reduction cell α_{reduce}. Cell structures are repeatedly stacked to form the final CNN structure. There are only two reduction cells in the final CNN structure, located in the 1/3 and 2/3 depths of the network. The best architecture searched by our iDARTS on the DARTS search space is obtained with $T = 4$ and $K = 2$. In CIFAR-10, we stack 20 cells to form the final structure for training. The batch size is set as 96, and the number of initial filters is 36. We then transfer the best-searched cells to CIFAR-100 and ImageNet to evaluate the transferability. The experiment setting for the evaluation in CIFAR-100 is the same as CIFAR-10. In the ImageNet dataset, the experiment setting is slightly different from CIFAR-10 in that only 14 cells are stacked, and the number of initial channels is changed to 48, and the batch size is set as 128. We use a linear learning rate scheduler and also following PDART (Chen et al., 2019a) and PCDARTS (Xu et al., 2020) to use a smaller slope in the last five epochs for the architecture evaluation on the ImageNet.

E. Comparison of methods to approximate the hypergradient

We compare different hypergradient approximations in Table 4, which summarizes the computational complexity and memory cost for each method. Under the assumption that Hessian vector products are computed with the *autograd*, we know that the compute time and memory cost for computing a Hessian vector product are with a constant factor of the compute time and memory used for computing a single derivative $\frac{\partial \mathcal{L}_2}{\partial w}$ (Rajeswaran et al., 2019; Griewank, 1993; Griewank & Walther, 2008). We denote that the memory cost for computing the gradient of supernet weight w and architecture parameters α are P and H, respectively. We consider each step in the **Steps** means the computational time of computing a Hessian vector product. The Conjugate Gradient considers iterative solver (e.g., CG) to calculate the inverse of Hessian, where S is the CG solver optimization steps, and each step contains the computation of Hessian vector product.