Stochastic Gradient Estimator for Differentiable NAS

author names withheld

Under Review for OPT 2022

Abstract

Neural architecture search (NAS) has recently attracted more attention due to its ability to design deep neural networks automatically. Differentiable NAS methods have predominated due to their search efficiency. However, differentiable NAS methods consistently adopt approximate gradient-based methods to solve bilevel optimization problems. While second derivative approximation optimizes Jacobian or/and Hessian vector computation, it is imprecise and time-consuming in practice. In this paper, we revisit the hypergradient of bilevel optimization problems in NAS, then propose a new optimizer based on a stochastic gradient estimator(SGE) for the computation of the Jacobian matrix in the hypergradient. The SGE is adaptable to previous differentiable NAS methods and more accurate than previously direct gradient approximation method. In the experiments on commonly differentiable NAS benchmarks, the proposed SGE-NAS algorithm outperforms the baseline algorithm. The test result demonstrates that the proposed SGE-NAS can effectively reduce search time and find the model with higher classification performance.

1. Introduction

Neural Architecture Search (NAS) is an effective method on automating the process of neural network design, with achieving remarkable success on various deep learning applications [1, 4, 10, 12]. A recently proposed method Differentiable ARchiTecture Search (DARTS) [6] adopts the continuous relaxation to convert the operation selection problem into the continuous magnitude optimization for a set of candidate operations, which makes it possible solve the bilevel optimization problem in NAS via continuous method [8].

Gradient-based methods are widely used in bilevel optimization problems for NAS [2, 6, 14, 15]. Hypergradient in bilevel optimization is the gradient of the outer variable and depends on the minimal solution of the inner function [8]. Previous differentiable NAS methods [6] commonly replace hypergradient by a simple approximation scheme. The architecture parameters α and network weights parameters w are alternately updated on the validation and training datasets via gradient descent, with approximating either the previous step forward or the current w^*. Only two forward passes to the weights and two backward passes to α are needed to compute the hypergradient, thus the complexity is partially reduced [6]. It also lays a hidden trouble for the common performance collapse problem of differentiable NAS [15]. However, due to the high dimension of the neural network parameter α, gradient-based methods are still computationally expensive in practice after approximate processing.

To overcome the computational challenges of current gradient-based methods, we employ an

© .

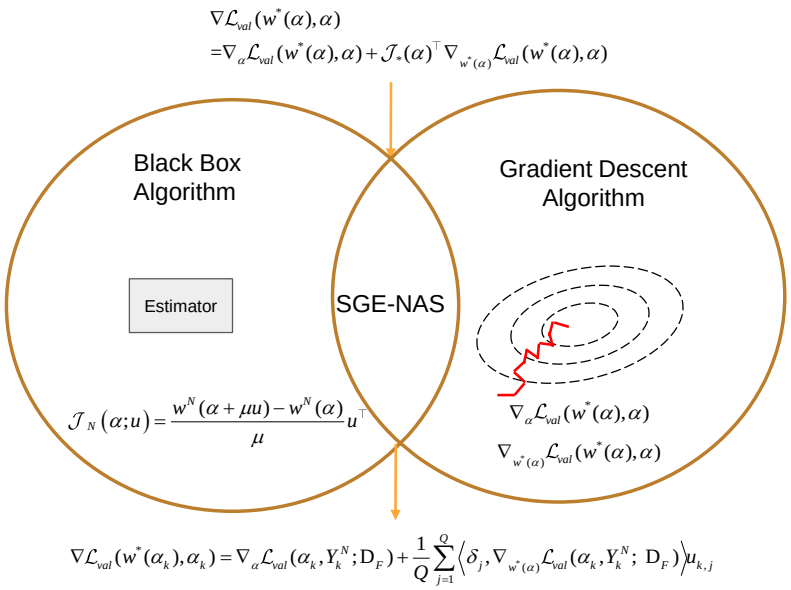

Figure 1: Overall on the calculation process of the hypergradient via our SGE-NAS

estimation approach to optimize the second-order computation namely stochastic gradient estimator(SGE). An ES(evolution strategy)-based bilevel optimizer is proposed for hyper-parameter optimization [3], which approximates the hypergradient, treating it completely as a black-box objective. However, bilevel optimization as Eq. 3 has a more complex form than the gradient in the standard minimization problem. SGE in our method is better combined with Eq. 3 and only estimates the Jacobian matrix, thus alleviate the error of the two-step gradient approximation and has better performance than the method that uses ES estimation.

Overall, our contributions to this work are summarized as followed:

In this paper, SGE-NAS is proposed, a novel differentiable NAS combined with estimator and gradient descent. SGE-NAS replaces the previous two-step approximation algorithm and improves the search accuracy.

A simple but effective stochastic gradient estimator(SGE) is introduced to optimize the complex Jacobian matrix calculation, which can be easily adapted with previous gradient-based methods to reduce the search cost.

2. Methods

2.1. Hypergradient of NAS

Differentiable NAS stacks the entire neural network by searching the structure of nomal cells and reduction cells [6]. Typically, a cell is defined as a Directed Acyclic Graph (DAG) with N nodes, where each node represents a potential connection and the information between every two nodes is transformed by an edge [13]. Each edge (i,j) contains multiple candidate operations, and the discrete operation selection is transformed into a differentiable parameter optimization problem by applying continuous relaxation through a learnable architecture parameter set α to mix the outputs of different operations.

$$\bar{O}^{(i,j)}(x)=\sum_{k=1}^{|\mathcal{O}|}\beta_k^{(i,j)}O_k(x) \quad \beta_k^{(i,j)}=\frac{\exp\left(\alpha_k^{(i,j)}\right)}{\sum_{k'=1}^{|\mathcal{O}|}\exp\left(\alpha_{k'}^{(i,j)}\right)} \tag{1}$$

where x and $\bar{O}$ are the input and mixed output of an edge, $\mathcal{O}$ is the candidate operation set, and β denotes the softmax-activated architecture parameter set. In this way, we can perform architecture search in a differentiable manner by solving following bilevel optimization objective.

$$\begin{aligned} &\min_{\alpha} \mathcal{L}_{val}\left(w^*(\alpha), \alpha\right) \\ &\text{s.t. } w^*(\alpha) = \arg\min_{w} \mathcal{L}_{train}(w, \alpha) \end{aligned} \tag{2}$$

Since the optimization of the inner variable α is too expensive, it is impossible to accurately evaluate the architectural gradient. In the common-used gradient-based bilevel optimizer, the key step is the estimation of the hypergradient.Different from the approximation processing step widely used in gradient-based NAS[9]. The hypergradient of the optimization objective in Eq. 2 is expanded and analyzed in the following form

$$\begin{aligned} &\nabla \mathcal{L}_{val}(w^*(\alpha), \alpha) \\ =&\nabla_{\alpha} \mathcal{L}_{val}(w^*(\alpha), \alpha) + \mathcal{J}_*(\alpha)^{\top} \nabla_{w^*(\alpha)} \mathcal{L}_{val}(w^*(\alpha), \alpha) \end{aligned} \tag{3}$$

where the Jacobian matrix $\mathcal{J}_*(\alpha) = \frac{\partial w^*(\alpha)}{\partial \alpha}$ including the Hessian inverse and the second-order mixed derivative, and expansed as

$$\mathcal{J}_*(\alpha) = -\left[\nabla^2_{w^*(\alpha)} \mathcal{L}_{\text{train}}\left(\alpha, w^*(\alpha)\right)\right]^{-1} \nabla_{\alpha} \nabla_{w^*(\alpha)} \mathcal{L}_{\text{train}}\left(\alpha, w^*(\alpha)\right) \tag{4}$$

Eq. 3 consists of two parts, direct gradient $\nabla_{\alpha} \mathcal{L}_{val}(w^*(\alpha), \alpha)$ and indirect gradient $\mathcal{J}_*(\alpha)^{\top}$ $\nabla_{w^*(\alpha)} \mathcal{L}_{val}(w^*(\alpha), \alpha)$. The direct component can be efficiently computed using existing automatic differentiation techniques from deep learning. However, the indirect component is much more computationally complex.

2.2. Stochastic gradient estimator

In this section, we propose a new estimation method to solve the computational problem of hypergradient. An estimation method [3] takes the objective function as a black-box objective and directly applies the ES (Evolution Strategy) method to estimate the hypergradient of the objective function. Because the hypergradient form is complex and very sensitive, such estimation may lead to a large bias error [5, 7]. Instead of this approach, the gradient $\nabla_{\alpha} \mathcal{L}_{val}(w^*(\alpha), \alpha)$ and $\nabla_{w^*(\alpha)} \mathcal{L}_{val}(w^*(\alpha), \alpha)$ can be computed efficiently and accurately in Eq. 3, while the Jacobian matrix $\mathcal{J}_*(\alpha)$ is the main term causing computational difficulties. We subsume the Hessian matrix into $\mathcal{J}_*(\alpha)$, then use the stochastic gradient estimator (SGE) to form a Hessian-free method without two-step approximation, which makes the calculation of the hypergradient more accurate and time-saving. The overall bilevel optimization process is shown in Algorithm 1.

Specifically, our SGE-NAS optimizer has the following two steps:

Firstly, for the inner variable α: The amplitude is set to $\alpha_k + \mu u_{k,j}$, and all SGD processes are sampled in the same size batch $\{\mathrm{S}_0, ..., \mathrm{S}_{N-1}\}$. Inspired by [11], $\widehat{\mathcal{J}}_N(\alpha_k; u_{k,j})$ can be calculated as

$$\widehat{\mathcal{J}}_N\left(\alpha; u\right) = \frac{w^N(\alpha + \mu u) - w^N(\alpha)}{\mu} u^{\top} \tag{5}$$

where u is an independent and identically distributed Gaussian vector. For any vector v, the vector product of the Jacobian can be efficiently computed using the dot product as $\widehat{\mathcal{J}}_N(\alpha;u)^\top v = \langle \delta(\alpha;u), v\rangle\, u$, where $\delta(\alpha;u) = \frac{y^N(\alpha+\mu u)-w^N(\alpha)}{\mu}$.

Secondly, for the inner variable $w^*(\alpha)$: Sample a new batch D_F independent from $\{\mathrm{S}_0, ..., \mathrm{S}_{N-1}\}$, where sampled as an estimator of the stochastic gradient $\nabla_\alpha \mathcal{L}_{val}(\alpha_k, Y_k^N; \mathrm{D}_F)$ and $\nabla_{w^*(\alpha)} \mathcal{L}_{val}(\alpha_k, Y_k^N; \mathrm{D}_F)$. The hypergradient is then estimated as

$$\nabla \mathcal{L}_{val}(w^*(\alpha_k), \alpha_k) = \nabla_\alpha \mathcal{L}_{val}(\alpha_k, Y_k^N; \mathrm{D}_F) + \frac{1}{Q}\sum_{j=1}^{Q} \left\langle \delta_j, \nabla_{w^*(\alpha)} \mathcal{L}_{val}(\alpha_k, Y_k^N;\ \mathrm{D}_F)\right\rangle u_{k,j} \quad (6)$$

Algorithm 1: Bilevel Optimizer with Stochastic Gradient Estimator

Data: inner and outer variables α and $w^*(\alpha)$, initializations α_0 and ${w_0}^*(\alpha)$, lower- and upper-level stepsizes p and q, inner and outer iterations numbers K and N, and number of Gaussian vectors Q.

Result: updated inner variable α_{k+1}, and hypergradient of bilevel optimization $\nabla \mathcal{L}_{val}(w^*(\alpha), \alpha)$

```
initialization;
while k = 0, 1, ..., K do
    set Y_k^0 = w_0*(α),   Y_{k,j}^0 = w_0*(α), j = 1, ..., Q ;
    generate u_{k,j} = N(0, I) ∈ R^p,   j = 1, ..., Q ;
    while t = 1, 2, ..., N do
        draw a sample batch S_{t-1} ;
        update Y_k^t = Y_k^{t-1} − p∇_{w*(α)} L_train(α_k, Y_k^{t-1}; S_{t-1}) ;
        while j = 1, 2, ..., Q do
            update Y_{k,j}^t = Y_{k,j}^{t-1} − p∇_{w*(α)} L_train(α_k + μu_{k,j}, Y_{k,j}^{t-1}; S_{t-1}) ;
        end
    end
    calculate δ_j = (Y_{k,j}^N − Y_k^N)/μ,   j = 1, ..., Q ;
    draw a sample batch D_F ;
    calculate ∇L_val(w*(α_k), α_k) via Eq. 6 ;
    update α_{k+1} = α_k − q∇L_val(w*(α), α) ;
end
```

3. Experiment

In this section, we conduct extensive experiments applying our SGE-NAS optimization algorithm based on the following differentiable NAS with SGD-Methods:(1) SGE-NAS-a: DARTS [6], (2) SGE-NAS-b: PC-DARTS [14],(3) SGE-NAS-c: Fair DARTS [2], (4) SGE NAS-d: β-DARTS [15].

Image benchmark datasets CIFAR-10 and CIFAR-100 are used in the experiments. The

Table 1: Performance comparison between SGE-NAS series and corresponding differentiable NAS on CIFAR-10 and CIFAR-100

Differentiable mode	Method	CIFAR-10			CIFAR-100		
		Top-1↑	Param.↓	Cost↓	Top-1↑	Param.↓	Cost↓
SGD	DARTS	97.06	3.3	9.6	79.48	1.8	9.6
	PC-DARTS	97.43	3.6	2.4	83.54	3.7	2.4
	Fair DARTS	97.46	3.1	9.6	83.62	3.2	9.6
	β-DARTS	97.49	3.7	9.6	83.76	3.8	9.6
SGE	SGE-NAS-a	97.65	3.3	6.5	82.98	1.9	6.5
	SGE-NAS-b	97.88	3.5	1.9	84.72	3.5	1.9
	SGE-NAS-c	97.59	3	7.1	84.69	3.1	7.1
	SGE-NAS-d	97.98	3.8	6.3	85.12	3.8	6.3

search space for the candidate operations includes: 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated separable convolutions, 3×3 max pooling, 3×3 average pooling, and skip connection. The hyperparameters for the architecture are kept the same as DARTS, PC-DARTS, Fair DARTS, and β-DARTS. SGD optimizer is set with a momentum of 0.9 and a weight decay of 3e-4, and the initial learning rate is set to 0.005 with a cosine decay scheduler. For our SGE optimizer, we set lower- and upper-level stepsizes p=0.01, q=0.01, number of Gaussian vectors Q=10 in Algorithm 1 for searching the best architecture.
Table 1 shows that all SGE-NAS series models achieve comparable or better performance than the baselines in terms of top-1 accuracy. Among them, models with comparable accuracy on CIFAR-10 are obtained while our search cost is 34.3% GPU hours less than β-DARTS [15]. Besides, there is an improvement of 1.05% $\sim$ 3.50% of top-1 accuracy on CIFAR-100 with approximately 20% $\sim$ 34% search time reduction. After easily adapting to the baselines, our SGE-NAS shows its advantage in reducing search time and improving model accuracy compared to previous gradient-based methods.

4. Conclusion

In this work, we propose SGE-NAS, a novel Neural Architecture Search optimization algorithm based on stochastic gradient estimation, to improve the search speed and reduce the approximation error. Specifically, we analyze the hypergradient of bilevel optimization in NAS and propose a stochastic gradient estimator for the Jacobian matrix to relieve the most time-consuming part. The proposed estimator is combined with the gradient descent of the derivative process to form a new method to solve the NAS bilevel optimization problem. Extensive experimental results on several benchmark datasets demonstrate the effectiveness and efficiency of the SGE-NAS algorithm, which keeps low search costs and achieves better search accuracy performance.

References

[1] Yukang Chen, Tong Yang, Xiangyu Zhang, Gaofeng Meng, Xinyu Xiao, and Jian Sun. Detnas: Backbone search for object detection. In Advances in Neural Information Processing Systems, pages 6642–6652, 2019.

[2] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. arXiv preprint arXiv:1907.01845, 2019.

[3] Bin Gu, Guodong Liu, Yanfu Zhang, Xiang Geng, and Heng Huang. Optimizing large-scale hyperparameters via automated learning algorithm. arXiv preprint arXiv:2110.07004, 2021.

[4] Yufan Jiang, Chi Hu, Tong Xiao, Chunliang Zhang, and Jingbo Zhu. 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), pages 3576–3581, 2019.

[5] Xiang Li, Chen Lin, Chuming Li, Ming Sun, Wei Wu, Junjie Yan, and Wanli Ouyang. Improving one-shot nas by suppressing the posterior fading. arXiv preprint arXiv:1910.02543, 2019.

[6] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. ICLR, 2019.

[7] Jelena Luketina, Mathias Berglund, Klaus Greff, and Tapani Raiko. Scalable gradient-based tuning of continuous regularization hyperparameters. In International conference on machine learning, pages 2952–2960, 2016.

[8] Gregory M Moore. Bilevel programming algorithms for machine learning model selection. Rensselaer Polytechnic Institute, 2010.

[9] Takayuki Okuno, Akiko Takeda, and Akihiro Kawana. Hyperparameter learning via bilevel nonsmooth optimization. arXiv preprint arXiv:1806.01520, 2018.

[10] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. arXiv preprint arXiv:2006.02903, 2020.

[11] Daouda Sow, Kaiyi Ji, and Yingbin Liang. A novel hessian-free bilevel optimizer via evolution strategies. arXiv preprint arXiv:2110.07004, 2022.

[12] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In International Conference on Machine Learning, pages 6105–6114, 2019.

[13] Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable nas. In International Conference on Learning Representations, 2021.

[14] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. In ICLR, 2020.

[15] Peng Ye, Baopu Li, Yikang Li, Tao Chen, Jiayuan Fan, and Wanli Ouyang. β-darts: Beta-decay regularization for differentiable architecture search. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 10864–10873, 2022.