SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

November 2023, Vol. 66 212101:1–212101:13
https://doi.org/10.1007/s11432-021-3527-7

Optimization-inspired manual architecture design and neural architecture search

Yibo YANG[1], Zhengyang SHEN[2], Huan LI[3] & Zhouchen LIN[4,5,6*]

[1]*JD Explore Academy, Beijing* 100176*, China;*
[2]*School of Mathematical Sciences, Peking University, Beijing* 100871*, China;*
[3]*Institute of Robotics and Automatic Information Systems, College of Artificial Intelligence, Nankai University, Tianjin* 300071*, China;*
[4]*Key Lab of General Artificial Intelligence, School of Intelligence Science and Technology, Peking University, Beijing* 100871*, China;*
[5]*Institute for Artificial Intelligence, Peking University, Beijing* 100871*, China;*
[6]*Pazhou Lab, Guangzhou* 510335*, China*

Received 8 June 2021/Revised 10 December 2021/Accepted 16 February 2022/Published online 5 September 2023

Abstract Neural architecture has been a research focus in recent years due to its importance in deciding the performance of deep networks. Representative ones include a residual network (ResNet) with skip connections and a dense network (DenseNet) with dense connections. However, a theoretical guidance for manual architecture design and neural architecture search (NAS) is still lacking. In this paper, we propose a manual architecture design framework, which is inspired by optimization algorithms. It is based on the conjecture that an optimization algorithm with a good convergence rate may imply a neural architecture with good performance. Concretely, we prove under certain conditions that forward propagation in a deep neural network is equivalent to the iterative optimization procedure of the gradient descent algorithm minimizing a cost function. Inspired by this correspondence, we derive neural architectures from fast optimization algorithms, including the heavy ball algorithm and Nesterov's accelerated gradient descent algorithm. Surprisingly, we find that we can deem the ResNet and DenseNet as special cases of the optimization-inspired architectures. These architectures offer not only theoretical guidance, but also good performances in image recognition on multiple datasets, including CIFAR-10, CIFAR-100, and ImageNet. Moreover, we show that our method is also useful for NAS by offering a good initial search point or guiding the search space.

Keywords deep neural network, manual architecture design, neural architecture search, image recognition, optimization algorithms, learning-based optimization

Citation Yang Y B, Shen Z Y, Li H, et al. Optimization-inspired manual architecture design and neural architecture search. Sci China Inf Sci, 2023, 66(11): 212101, https://doi.org/10.1007/s11432-021-3527-7

1 Introduction

Deep neural networks (DNNs) have been successfully applied in many areas, including image recognition [1], object detection [2, 3], and visual segmentation [4]. With the ImageNet [1] performance breakthrough, several deep network architectures have been designed. The classification results are gradually improved by increasing the network depth and capacity [5–7]. Highway networks [8] and residual networks (ResNets) [9] introduce the skip connections and successfully train very deep networks. Later, manual architecture design adopted more complex topologies and connections [10–12]. All these neural architectures were designed mainly based on empirical analyses and experimental observations. However, they may not be the optimal choice of architecture and have the best performance overhead trade-off, inspiring a series of explorations on neural architecture search (NAS) in the recent literature. Popular methods include reinforcement learning that assigns a good architecture on a validation set with a high reward [13–20], evolution algorithms [21–27], and gradient-based methods [28–34]. Despite the success of these architectures, neither manual architecture design nor NAS has a clear understanding of

* Corresponding author (email: zlin@pku.edu.cn)

deep network behaviors. How to develop theoretical guidance for neural architectures to benefit manual architecture design and NAS is the core challenge that we target in this study.

To improve the interpretability of neural architectures, some studies have attempted to build connections on areas with rich theories and analytical tools, such as numerical methods of ordinary differential equations (ODEs) [35–38] and optimization algorithms [39–44]. Specifically, the forward propagation of a neural network can be deemed as the iterative procedure of an optimization algorithm. The final layer of the network outputs an optimal solution $\boldsymbol{x}^*$ that minimizes a cost function, known as learning-based optimization. The architecture of learning-based optimization is transparent and explainable because it is translated from a well-developed algorithm in optimization theory. However, the current optimization-inspired architecture is only limited to solving optimization problems, such as compressed sensing. We cannot use these methods with known objective functions to construct architectures for general purposes, such as image recognition. If we can move a step forward to build the connection between optimization and the neural architecture used for general feature learning, then a broader range of neural architectures can be explainable. Moreover, the connection may inspire us to improve the manual architecture design and NAS following the guidance from optimization theory.

Motivated by the above analysis, in this study, we generalize the idea of learning-based optimization. We prove that the forward propagation in a network for image recognition also corresponds to a gradient-based optimization algorithm. Bridging the connection between the general neural architecture and optimization algorithm, we hypothesize that an optimization algorithm with a good convergence rate may imply a neural architecture with good performance. Accordingly, we develop new architectures inspired by traditional optimization algorithms. Our optimization-inspired manual architecture design not only has theoretical guidance but also performs better than traditional architectures. Moreover, our method is useful for NAS by offering a good initial search point or guiding the search space.

Our methodology is inspired by optimization algorithms and is applied to the manual architecture design and NAS, both of which lacked theoretical guidance before our study. Specifically, our contributions include the following.

(1) In the general case, we prove that the propagation in a DNN under certain conditions is equivalent to the iterative procedure of an optimization algorithm minimizing a cost function $F(\boldsymbol{x})$. As a comparison, the learning-based optimization architectures only adopt the soft thresholding operator as the nonlinear activation and can be used only for the compressed sensing problem.

(2) Inspired by the correspondence, we hypothesize that the optimization algorithm with a good convergence rate may imply a neural architecture with good performance. We derive neural architectures, HBNet and AGDNet, from the heavy ball (HB) and Nesterov's accelerated gradient algorithms, respectively. Two existing popular architectures are the special cases of our optimization-inspired architectures.

(3) Experimental results on multiple datasets, including CIFAR-10, CIFAR-100, and ImageNet, demonstrate that our optimization-inspired architectures are competitive with or even outperform their counterparts, i.e., ResNet and DenseNet. Furthermore, in the experiments of NAS, we adopt the HBNet architecture as the initial search point and use the connection pattern of HBNet and AGDNet to modify the search space. Both show improvements over the original settings at low costs. These results show that our optimization-inspired methodology is promising for manual architecture design and NAS.

2 Related work

Manual architecture design. Manual architecture design denotes neural architectures designed by humans based on their empirical understanding of tasks to which DNNs are applied. In the early stage of the manual architecture design, genetic algorithm [45, 46]-based approaches were popular. However, networks designed by genetic algorithms perform worse than only the designing architecture [47]. Ref. [48] proposed to spare the effort of architecture selection. Bayesian optimization [49] and meta-modeling [50] are used to determine the operations and hyperparameters. Some adaptive strategies were adopted to design the whole architecture layer by layer [51–53]. In recent years, manual architecture design has been popular and revolutionized a wide range of computer vision tasks [1, 5–10, 12], but designing architectures suffer from relying on empirical knowledge and exhaustive engineering trials.

NAS. Because manual architecture design cannot ensure an optimal architecture, recent studies have used the automated machine learning technique to automatically produce satisfactory architectures, also known as NAS. The key difference between the manual architecture design and NAS is that the design

needs all detailed human specifications on the architecture, and the search only requires humans to provide an overall search space and then computer searches for more details. The current studies can be categorized into three frameworks. The reinforcement learning search method introduces an agent to generate architectures and assign a good one with a high reward [13, 14]. Follow-up studies focus on proposing a good search space or algorithm to reduce the search cost [18–20]. Evolution-based searching uses evolutionary algorithms for optimizing neural architectures [21–27]. However, the computation and time required by these methods are still not acceptable. Another line of search is one-shot-based methods that significantly reduce the search cost by treating all potential architectures as subgraphs of a supernetwork (supernet) and sparing the efforts of evaluating each candidate [16, 29–34, 54–60]. In differentiable architecture search (DARTS) [29], the search space is relaxed to be differentiable, so the supernet is jointly trained with architecture variables. Despite its simplicity, DARTS-based methods suffer from instability and an architectural gap between the search and evaluation. Some later studies adopted the Gumbel-softmax strategy to reduce the architectural gap [30, 34]. In [31], a progressive shrinking method is proposed to bridge the depth gap between the search and evaluation. DenseNAS [33] and partially connected DARTS [32] introduce more trainable parameters to take path probabilities into account. A recent study formulated NAS as a sparse coding problem [61]. However, these search methods lack theoretical guidance. Similar to manually designed ones, the searched architectures are not interpretable either. In this study, we show that our optimization-inspired methodology is beneficial to NAS with clear guidance.

Optimization-inspired networks. To make a theoretical guidance for neural architecture and improve the interpretability, researchers have built connections between neural networks and some well-developed areas, such as numerical methods of ODEs and optimization methods, to analyze and specify neural architectures using their rich analytical tools. In particular, optimization-inspired networks refer to architectures designed from the optimization perspective. The early optimization-inspired network [39] was derived by unrolling the traditional iterative optimization method, iterative shrinkage-thresholding algorithm (ISTA) [62], which iterates as $\boldsymbol{x}_{k+1} = \text{prox}_{\frac{\lambda}{L}\|\cdot\|_1}(\boldsymbol{x}_k - \frac{1}{L}\boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y}))$, where $\text{prox}_{\lambda\|\cdot\|_1}(\boldsymbol{x}) = \text{argmin}_{\boldsymbol{z}} \frac{1}{2}\|\boldsymbol{z}-\boldsymbol{x}\|^2+\lambda\|\boldsymbol{z}\|_1$, to solve the compressed sensing problem defined as $\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{y}\|^2+\lambda\|\boldsymbol{x}\|_1$[1)]. It aims to train a deep network where learnable weights $\boldsymbol{W}_k$ replace the fixed transformation matrix $\boldsymbol{A}$ in the iterative optimization method, the nonlinear activation is achieved by the proximal operator, and the connection pattern corresponds to the calculation of each iteration. Hence, the architecture is decided by the optimization method. Compared with the traditional iteration-based ISTA, the learning-based ISTA (LISTA) enjoys a better convergence speed and inspires a series of later studies [40–43, 63, 64]. In [65], a theoretical understanding was offered, and the linear convergence of LISTA was proven. Apart from the unrolling ISTA, many studies have also unrolled other optimization algorithms, such as iterative hard thresholding [40], approximate message passing [66], and alternating direction method of multipliers (ADMMs) [43, 44].

Although these optimization-inspired networks have theoretical guidance and achieved great success, they can only be used to solve given optimization problems, such as sparse coding and compressed sensing for image reconstruction, instead of general feature learning tasks, such as image recognition. Our preliminary study [67][2)] aimed at generalizing the optimization-inspired network for image reconstruction into the general feature learning purpose. It established the connection between a fast optimization algorithm and its corresponding good neural architecture and proposed to design architectures based on this connection. However, it only focused on the manual architecture design. In this study, we move a step forward to generalize this idea. We show that our methodology can also benefit NAS in two ways, i.e., using the optimization-inspired networks as the initial search point or using the optimization-inspired connection pattern as a better search space. It brings theoretical interpretability to the searched architecture and improves the search results.

Concretely, the improvements and differences of this study over our conference version can be listed as follows.

(1) We rewrite and polish the contents that have been in the conference version. We add more experiments on the optimization-inspired networks on the ImageNet dataset in Section 6 of this journal version.

(2) We extend our optimization-inspired methodology from the manual architecture design to NAS.

1) We denote $\|\boldsymbol{x}\| = \sqrt{\sum_i \boldsymbol{x}_i^2}$ and $\|\boldsymbol{x}\|_1 = \sum_i |\boldsymbol{x}_i|$.

2) The preliminary version of this paper has appeared on the ACML 2018 conference, and a patent has been filed.

Our method is also useful for NAS in two ways, i.e., offering a good initial search point and guiding the search space.

(3) We conduct experiments of NAS on the CIFAR and ImageNet datasets to demonstrate our method in Section 7. We show that the optimization-inspired networks, HBNet and AGDNet, correspond to better search spaces, where better architectures can be searched from. The HBNet itself can also be used as a good initial search point to make the search process more efficient. By contrast, our conference version only focused on the optimization-inspired manual architecture design and did not contribute to NAS.

(4) Our experiments, including the original results in the conference version and the newly added results of the manual architecture design and NAS indicate that the optimization-inspired methodology can serve as a good way to design or search for high-performance neural architectures, which is much easier and more interpretable than manually designing from scratch or searching without any guidance. Based on our all contributions, we further conclude that the connection between the optimization algorithm and DNN architecture has been empirically validated, and the proposed methodology can offer good guidance for manual architecture design and NAS, which is beyond our conference version that only validates the benefit on manual architecture design with limited experimental results on ImageNet.

3 Preliminaries on optimization algorithms

We briefly review some optimization algorithms, including the gradient descent (GD) algorithm [68], the HB algorithm [69], the Nesterov's accelerated gradient descent (AGD) algorithm [70], and the ADMM algorithm [71, 72].

The GD algorithm with a fixed stepsize[3)] of 1 iterates as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \nabla f(\boldsymbol{x}_k). \tag{1}$$

The HB algorithm adds a momentum after GD and iterates as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \nabla f(\boldsymbol{x}_k) + \beta(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}). \tag{2}$$

The AGD algorithm introduces the momentum mechanism as follows:

$$\begin{aligned} \boldsymbol{y}_k &= \boldsymbol{x}_k + \frac{\theta_k(1-\theta_{k-1})}{\theta_{k-1}}(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}), \\ \boldsymbol{x}_{k+1} &= \boldsymbol{y}_k - \nabla f(\boldsymbol{y}_k), \end{aligned} \tag{3}$$

where θ_k satisfies that $\frac{1-\theta_k}{\theta_k^2} = \frac{1}{\theta_{k-1}^2}$ and $\theta_0 = 1$. When $f(\cdot)$ is μ-strongly convex i.e., $f(\boldsymbol{x}_2) \geqslant f(\boldsymbol{x}_1) + \langle \nabla f(\boldsymbol{x}_1), \boldsymbol{x}_2 - \boldsymbol{x}_1 \rangle + \frac{\mu}{2}\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|^2$, and its gradient is L-Lipschitz continuous, i.e., $\|\nabla f(\boldsymbol{x}_2) - \nabla f(\boldsymbol{x}_1)\| \leqslant L\|\boldsymbol{x}_2 - \boldsymbol{x}_1\|$, $\frac{\theta_k(1-\theta_{k-1})}{\theta_{k-1}}$ is fixed at $\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}$, and the HB and AGD algorithms are able to find an ϵ-accuracy solution in $O\big(\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon}\big)$ iterations. As a comparison, the GD algorithm takes $O\big(\frac{L}{\mu}\log\frac{1}{\epsilon}\big)$ iterations. Iteration (3) can be reformulated as follows:

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k - \sum_{j=0}^{k} h_{k+1,j}\nabla f(\boldsymbol{y}_j), \tag{4}$$

where

$$h_{k+1,j} = \begin{cases} \dfrac{\theta_{k+1}(1-\theta_k)}{\theta_k} h_{k,j}, & j = [0, 1, \ldots, k-2], \\ \dfrac{\theta_{k+1}(1-\theta_k)}{\theta_k}(h_{k,k-1} - 1), & j = k-1, \\ 1 + \dfrac{\theta_{k+1}(1-\theta_k)}{\theta_k}, & j = k. \end{cases} \tag{5}$$

3) This can be easily achieved by scaling $f(\boldsymbol{z})$ such that the Lipschitz constant of $\nabla f(\boldsymbol{z})$ is 1.

The ADMM algorithm and its linearized version can also solve the problem: $\min_{\boldsymbol{y},\boldsymbol{x}} f(\boldsymbol{x})+f(\boldsymbol{y})$, s.t. $\boldsymbol{y}-\boldsymbol{x}=0$. With the penalty parameter fixed as 1, the Linearized ADMM algorithm is composed of the following steps:

$$\begin{aligned}
&\boldsymbol{x}_{k+1}=\operatorname{argmin}_{\boldsymbol{x}}\langle\nabla f(\boldsymbol{x}_k),\boldsymbol{x}\rangle+\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{x}_k\|^2+\langle\boldsymbol{\lambda}_k,\boldsymbol{x}\rangle+\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}_k\|^2,\\
&\boldsymbol{y}_{k+1}=\operatorname{argmin}_{\boldsymbol{y}}\langle\nabla f(\boldsymbol{y}_k),\boldsymbol{y}\rangle+\frac{1}{2}\|\boldsymbol{y}-\boldsymbol{y}_k\|^2-\langle\boldsymbol{\lambda}_k,\boldsymbol{y}\rangle+\frac{1}{2}\|\boldsymbol{x}_{k+1}-\boldsymbol{y}\|^2,\\
&\boldsymbol{\lambda}_{k+1}=\boldsymbol{\lambda}_k+(\boldsymbol{x}_{k+1}-\boldsymbol{y}_{k+1}),
\end{aligned}\tag{6}$$

where $\boldsymbol{\lambda}$ is the Lagrange multiplier.

4 Modeling the propagation in feedforward neural network

A DNN propagates as

$$\boldsymbol{z}_{k+1}=\Phi(\boldsymbol{W}_k\boldsymbol{z}_k),\tag{7}$$

where $\boldsymbol{W}_k$ denotes the linear transformation, $\boldsymbol{z}_k$ represents the k-th layer feature, and Φ is the non-linear activation, e.g., the ReLU or sigmoid functions. We fix the matrices $\boldsymbol{W}_k$ as $\boldsymbol{W}$ to simplify the analysis.

Lemma 1. Assuming that $\boldsymbol{W}$ is symmetric and positive definite[4)], we have that there exists a function $f(\boldsymbol{z})$ such that Eq. (7) is equivalent to minimizing $F(\boldsymbol{z})=f(\boldsymbol{U}\boldsymbol{z})$, where $\boldsymbol{U}=\sqrt{\boldsymbol{W}}$, with the following steps:

(1) Define a new variable $\boldsymbol{x}=\boldsymbol{U}\boldsymbol{z}$;

(2) Use (1) to minimize $f(\boldsymbol{x})$;

(3) Recover $\boldsymbol{z}_0,\boldsymbol{z}_1,\ldots,\boldsymbol{z}_k$ from $\boldsymbol{x}_0,\boldsymbol{x}_1,\ldots,\boldsymbol{x}_k$ via $\boldsymbol{z}=\boldsymbol{U}^{-1}\boldsymbol{x}$.

Proof. Please refer to [67] for the proof.

The objective functions $f(\boldsymbol{z})$ for most widely adopted non-linear activations are listed in Appendix A.

5 Extension to other optimization algorithms

In Section 4, we have shown that the propagation in a DNN can be viewed as the optimization procedure of minimizing a cost function $F(\boldsymbol{z})$ using the GD algorithm. We consider the case where other optimization algorithms are used to minimize $F(\boldsymbol{z})$ in this section.

The HB algorithm. First, we consider the HB algorithm, i.e., iteration (2). Similarly, we can minimize $F(\boldsymbol{z})=f(\boldsymbol{U}\boldsymbol{z})$ with the following three steps:

(1) Variable substitution $\boldsymbol{x}=\boldsymbol{U}\boldsymbol{z}$.

(2) Use (2) to minimize $f(\boldsymbol{x})$, which is defined as

$$f(\boldsymbol{x})=\frac{\|\boldsymbol{x}\|^2}{2}-\sum_i\Psi(\boldsymbol{U}_i^{\mathrm{T}}\boldsymbol{x}),\tag{8}$$

where $\boldsymbol{U}_i$ denotes the i-th column of $\boldsymbol{U}$. So we have

$$\nabla f(\boldsymbol{x}_k)=\boldsymbol{x}_k-\boldsymbol{U}\Phi(\boldsymbol{U}\boldsymbol{x}_k).\tag{9}$$

And then Eq. (2) becomes

$$\boldsymbol{x}_{k+1}=\boldsymbol{x}_k-\nabla f(\boldsymbol{x}_k)+\beta(\boldsymbol{x}_k-\boldsymbol{x}_{k-1})=\boldsymbol{U}\Phi(\boldsymbol{U}\boldsymbol{x}_k)+\beta(\boldsymbol{x}_k-\boldsymbol{x}_{k-1}),$$

where we use (9) in the second equation.

(3) Recovering $\boldsymbol{z}$ from $\boldsymbol{x}$ via $\boldsymbol{z}=\boldsymbol{U}^{-1}\boldsymbol{x}$:

$$\begin{aligned}
\boldsymbol{z}_{k+1}&=\boldsymbol{U}^{-1}\boldsymbol{x}_{k+1}\\
&=\Phi(\boldsymbol{U}\boldsymbol{x}_k)+\beta(\boldsymbol{U}^{-1}\boldsymbol{x}_k-\boldsymbol{U}^{-1}\boldsymbol{x}_{k-1})\\
&=\Phi(\boldsymbol{U}^2\boldsymbol{z}_k)+\beta(\boldsymbol{z}_k-\boldsymbol{z}_{k-1})\\
&=\Phi(\boldsymbol{W}\boldsymbol{z}_k)+\beta(\boldsymbol{z}_k-\boldsymbol{z}_{k-1}).
\end{aligned}\tag{10}$$

4) $\boldsymbol{W}$ will be relaxed to be learnable in practical implementations.

Table 1 Error rates (%) of classification on CIFAR-10 and CIFAR-100 with and without data augmentation[a)]

Model	Params	CIFAR-10	CIFAR-100	CIFAR-10(+)	CIFAR-100(+)
Network in network [75]	–	10.41	35.68	8.81	–
All-CNN [76]	–	9.08	–	7.25	33.71
Deeply supervised net [77]	–	9.69	–	7.97	34.57
Highway network [8]	–	–	–	7.72	32.39
ResNet ($n = 9$)	0.85M	10.05	39.65	5.32	26.03
HBNet ($n = 9$)	0.85M	10.17	38.52	5.46	26
ResNet ($n = 18$)*	1.7M	9.17	38.13	5.06	24.71
HBNet ($n = 18$)	1.7M	8.66	36.4	5.04	23.93
DenseNet ($k = 12, L = 40$)	1.0M	7	27.55	5.24	24.42
AGDNet ($k = 12, L = 40$)	1.0M	6.44	26.33	5.2	24.87
DenseNet ($k = 12, L = 52$)	1.4M	6.05	26.3	5.09	24.33
AGDNet ($k = 12, L = 52$)	1.4M	5.75	24.92	4.94	23.84

a) "+" means that the standard data augmentation is enabled. "*" denotes that deep ResNet is not stable on CIFAR, so we implement multiple times and report the converged result, while our HBNet needs training only once.

The AGD algorithm. Then we consider the AGD algorithm, i.e., iteration (3). Similarly, we have

$$\boldsymbol{z}_{k+1} = \Phi\left(\boldsymbol{W}\left(\boldsymbol{z}_k + \beta_k(\boldsymbol{z}_k - \boldsymbol{z}_{k-1})\right)\right), \tag{11}$$

where $\beta_k = \frac{\theta_k(1-\theta_{k-1})}{\theta_{k-1}}$. We can also use the iteration (4) for $F(\boldsymbol{x})$

$$\boldsymbol{z}_{k+1} = \sum_{j=0}^{k} h_{k+1,j}\Phi(\boldsymbol{W}\boldsymbol{z}_j) + \boldsymbol{z}_k - \sum_{j=0}^{k} h_{k+1,j}\boldsymbol{z}_j. \tag{12}$$

The ADMM algorithm. Finally, we use the ADMM algorithm, i.e., iteration (6), for $F(\boldsymbol{x})$, which leads to the following steps:

$$\begin{aligned} \boldsymbol{z}'_{k+1} &= \frac{1}{2}\left(\Phi(\boldsymbol{W}\boldsymbol{z}'_k) + \boldsymbol{z}_k - \sum_{t=1}^{k}(\boldsymbol{z}'_t - \boldsymbol{z}_t)\right), \\ \boldsymbol{z}_{k+1} &= \frac{1}{2}\left(\Phi(\boldsymbol{W}\boldsymbol{z}_k) + \boldsymbol{z}'_{k+1} + \sum_{t=1}^{k}(\boldsymbol{z}'_t - \boldsymbol{z}_t)\right). \end{aligned} \tag{13}$$

We compare (10)–(13) with (7), and observe that the basic operation, $\Phi(\boldsymbol{W}\boldsymbol{z})$, is kept for all algorithms, but some additional side paths are introduced for the more advanced optimization algorithms, HB, AGD, and ADMM. It inspires our manual architecture design in Appendix C. The architecture derived by optimization algorithm X is referred to X-inspired architecture in our paper.

Based on the correspondence established above, we propose a conjecture that a faster optimization algorithm may imply a better neural network in Appendix B. Furthermore, we develop some optimization-inspired networks, HBNet and AGDNet, as detailed in Appendix C.

6 Experiments on manual architecture design

For fair comparison, we train all of our models using the training strategies adopted in DenseNet [10]. The Xavier initialization [73] for fully connected layers and the initialization method [74] for other layers are adopted. Table 1 [8, 75–77] shows the experimental results of manual architecture design on CIFAR. We conduct experiments for ResNet based models when the parameter n equals 9 and 18, in which cases the depth is 56 and 110, respectively. We conduct experiments for DenseNet based models when the growth rate k is 12 and the depth L is 40 and 52. As shown in Table 1, both HBNet and AGDNet achieve better performances than their corresponding baselines, i.e., ResNet and DenseNet, respectively. For AGDNet with k of 12 and L of 40, we achieve consistent improvements except for CIFAR-100 with augmentation. For AGDNet ($k = 12$ and $L = 52$), the improvements of our method are obvious on all datasets. Similarly, the improvements of HBNet over ResNet also grow larger as the depth increases. More importantly, as reported in ResNet [9], for very deep architecture, such as $n = 18$, a warm-up strategy is necessary for

Table 2 Error rates (%) of classification on ImageNet. We compare our optimization-inspired architectures with baselines of the same depth

Model	Top-1 (%)	Top-5 (%)	Params (M)	FLOPs (G)
ResNet-34	26.73	8.74	21.80	3.66
HBNet-34	26.33	8.56	21.80	3.66
ResNet-50	24.24	7.29	25.56	3.86
HBNet-50	23.93	7.06	25.56	3.87
ResNet-101	22.42	6.46	44.55	7.58
HBNet-101	21.91	6.07	44.55	7.59
DenseNet-121	25.02	7.71	7.98	2.88
AGDNet-121	24.62	7.39	7.98	2.92

a converged training. Indeed, our experiment using ResNet ($n = 18$) requires multiple trials to achieve a converged result. As a comparison, the training of our HBNet is more stable and is able to produce a converged result with training only once. Therefore, the optimization algorithm-inspired architecture, HBNet, is more stable to train when the depth is large.

The results of our proposed HBNet and AGDNet on ImageNet are shown in Table 2. Both HBNet and AGDNet achieve better classification performances than their baseline methods with similar parameter and computational cost. All these experimental results indicate that our optimization-inspired manual architecture design is very promising.

7 Experiments on NAS

Recently, DARTS has been widely used for NAS. But still, DARTS-based methods suffer from instability and the gap between the performance of the architecture in search and that in evaluation. In our work, we show that our optimization-inspired architectures can also be utilized to improve the search results of DARTS. Concretely, we will show that HBNet can inspire a search space that reduces the time consumption and can be a good initial search point to help search for a better architecture, while AGDNet inspires us to exploit a general and better search space.

In DARTS, a standard convolution cell is composed of $N = 7$ nodes. The output node c_k is defined as the depthwise concatenation of all the intermediate nodes (except for the input nodes), i.e.,

$$c_k = [x_0, x_1, x_2, x_3], \tag{14}$$

where $[\cdot]$ denotes the concatenation. The whole architecture is constructed by stacking the multiple cells. The first and second nodes of cell k are set equal to the outputs of cell $k-2$ and cell $k-1$, i.e., c_{k-2} and c_{k-1}, respectively. 1×1 convolutions are inserted when necessary.

7.1 Search space and initialization inspired by HBNet

We examine a simple search space that is consistent with HBNet, where the intermediate node x_t is computed based on its two predecessors, and the specific expression is

$$x_t = o_1^{(t-1,t)}(x_{t-1}) + o_2^{(t-1,t)}(x_{t-1}) + o^{(t-2,t)}(x_{t-2}), \tag{15}$$

where $o_1^{(t-1,t)}$ and $o_2^{(t-1,t)}$ denote the two operations applied to x_{t-1}, and $o^{(t-2,t)}$ denotes the operation applied to x_{t-2}. For ease of presentation, we denote c_{k-2} and c_{k-1} as x_{-2} and x_{-1} as well, respectively. The diagram for the HB-inspired search space is shown in Figure 1.

We use $\mathcal{O}$ to denote a set of 9 candidate operations, each of which is an operation $o(\cdot)$ to be applied to x, and the candidate operations are listed in Table 3. To make the search space differentiable, the choice of a particular operation $o(\cdot)$ from node i to j is relaxed to a softmax over all possible operations:

$$\overline{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} w_o^{(i,j)} o(x), \tag{16}$$

where

$$w_o^{(i,j)} = \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})}, \tag{17}$$

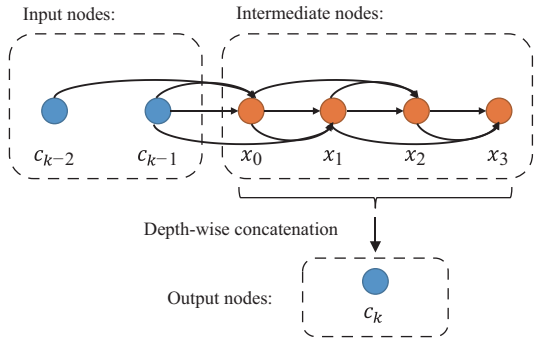

Figure 1 The diagram for the HB-inspired search space.

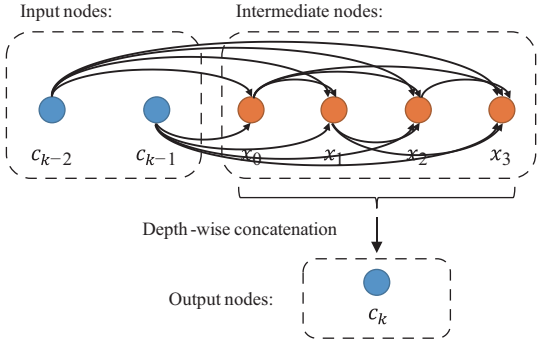

Figure 2 The diagram for the AGD-inspired search space.

Table 3 Candidate operations

Number	Candidate operations
1	3×3 separable conv
2	5×5 separable conv
3	3×3 dilated conv
4	5×5 dilated conv
5	3×3 max pooling
6	3×3 average pooling
7	Identity
8	Negative identity
9	Zero

and the weights to mix operations for a pair of nodes (i, j) are parameterized by a vector $\boldsymbol{\alpha}^{(i,j)}$ of dimension $|\mathcal{O}|$. The task of architecture search then reduces to learning a set of continuous variables $\boldsymbol{\alpha}^{\mathrm{HB}} = \{\boldsymbol{\alpha}_1^{(t-1,t)}, \boldsymbol{\alpha}_2^{(t-1,t)}, \boldsymbol{\alpha}^{(t-2,t)}\}$, and we name this HB-inspired search space as search space-HB in experiments.

Our optimization-inspired initialization particularly encodes the HBNet into the architecture parameters $\boldsymbol{\alpha}$ at the beginning of the architecture search. To be specific, we initialize the coefficient vectors $\boldsymbol{w}_2^{(t-1,t)}$ for $\overline{o}_2^{(t-1,t)}$ and $\boldsymbol{w}^{(t-2,t)}$ for $\overline{o}^{(t-2,t)}$ by constructing one-hot vectors. For $\overline{o}_2^{(t-1,t)}$, the coefficient for identity operation is 1; for $\overline{o}^{(t-2,t)}$, the coefficient for negative operation is 1. We initialize $\boldsymbol{w}_1^{(t-1,t)}$ for $\overline{o}_1^{(t-1,t)}$ using the softmax of a randomly initialized set of variables $\boldsymbol{\alpha}_{\mathrm{HBinit}} = \{\boldsymbol{\alpha}_{\mathrm{HBinit}}^{(t-1,t)}\}$. In this way, our optimization-inspired initialization leads to the following propagation:

$$x_t = F(x_{t-1}) + x_{t-1} - x_{t-2}, \tag{18}$$

where $F(\cdot)$ is a linear combination of 9 candidate operations listed in Table 3, and this is consistent with the HBNet architecture. Noting that in Appendix C.1, we specifically take $F(\cdot)$ as a composite function including two convolution layers, which restricts the HBNet architecture. By contrast, the architecture given in (18) is more flexible and general.

As a comparable baseline, all the parameters are randomly initialized by $\boldsymbol{\alpha}_{\mathrm{init}} = \{\boldsymbol{\alpha}_{1,\mathrm{init}}^{(t-1,t)}, \boldsymbol{\alpha}_{2,\mathrm{init}}^{(t-1,t)}, \boldsymbol{\alpha}_{\mathrm{init}}^{(t-2,t)}\}$.

7.2 Search space inspired by AGDNet

As for AGDNet, each intermediate node is computed based on its all predecessors. Using DARTS, this kind of propagation is always simply described as

$$x_t = \sum_{i<t} o^{(i,t)}(x_i). \tag{19}$$

Similarly, we use $\overline{o}(\cdot)$ defined in (16) to make the search space differentiable, and the search space becomes $\boldsymbol{\alpha}^{\mathrm{DARTS}} = \{\boldsymbol{\alpha}^{(i,j)}, i < j\}$.

However, different from the conventional DARTS, the intermediate node in (C6) is actually calculated based on its predecessors with the linear weighted sum method, i.e.,

$$x_t = \sum_{i<t} p_{it} o^{(i,t)}(x_i), \tag{20}$$

where p_{it} are the weight coefficients. Particularly, some operations like identity and negative identity that are explicitly expressed in (C6) are omitted because they are involved in the relaxed operation $\overline{o}(\cdot)$.

Considering that the number of the predecessors changes across iterations, which could cause undesired fluctuation in the resultant neural architecture, we introduce edge normalization to mitigate this problem. Then, the coefficients p_{it} become path probabilities, and they are computed using a softmax function over paths to node t:

$$p_{it} = \frac{\exp(\beta_{it})}{\sum_{j<t} \exp(\beta_{jt})}, \tag{21}$$

and we denote the introduced parameters as $\boldsymbol{\beta}^{\mathrm{AGD}} = \{\beta_{ij}, i < j\}$. As a result, the task of architecture search is modified to learning a larger set of variables $\boldsymbol{\Theta}^{\mathrm{AGD}} = \{\boldsymbol{\alpha}^{\mathrm{DARTS}}, \boldsymbol{\beta}^{\mathrm{AGD}}\}$, which is inspired by AGDNet. We name this AGD-inspired search space as search space-AGD in experiments, and the diagram for the AGD-inspired search space is shown in Figure 2.

7.3 Search algorithm and architecture evaluation

We evaluate our methods on CIFAR-10. To carry out architecture search, we randomly set 25000 training samples as the training set and the other 25000 training samples as the validation set. We denote the training and the validation loss as $\mathcal{L}_{\mathrm{train}}$ and $\mathcal{L}_{\mathrm{val}}$, respectively. We optimize the model weights by descending $\nabla_w \mathcal{L}_{\mathrm{train}}(\boldsymbol{w}, \boldsymbol{\alpha})$ or $\nabla_w \mathcal{L}_{\mathrm{train}}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ on the training set, and optimize the architecture parameters by descending $\nabla_{\boldsymbol{\alpha}} \mathcal{L}_{\mathrm{val}}(\boldsymbol{w}, \boldsymbol{\alpha})$ or $\nabla_{\boldsymbol{\alpha},\boldsymbol{\beta}} \mathcal{L}_{\mathrm{val}}(\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ on the validation set. We train a small network with 8 cells for 50 epochs, with batch size 64 (for both the training and validation sets) and the initial number of channels is 16. We use momentum SGD to optimize the weights $\boldsymbol{w}$, with initial learning rate $\eta_w = 0.025$ (annealed down to zero following a cosine schedule without restart [78]), momentum 0.9, and weight decay 3×10^{-4}. We use zero initialization for architecture variables $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, and use Adam [79] as the optimizer with initial learning rate $\eta_\alpha = \eta_\beta = 3 \times 10^{-4}$, momentum $(0.5, 0.999)$ and weight decay 10^{-3}.

When the searching is finished, for search space-HB, we retain the top-3 strongest non-zero operations from x_{k-1} and x_{k-2} to x_k to derive the final architecture. For search space-AGD, we retain the top-2 strongest non-zero operations from all previous nodes to x_k.

To evaluate the selected architecture, we randomly initialize its weights (weights learned during the search process are discarded), train it from scratch, and report its performance on the test set. Specifically, we train a large network of 20 cells for 600 epochs with batch size 96. The initial number of channels is increased from 16 to 36. Other hyperparameters remain the same as the ones used for architecture search. Additional enhancements include cutout [80], path dropout of probability 0.2, and auxiliary heads with weight 0.4. Note that the test set is never used for architecture search or architecture selection. The search cost and the test error are listed in Table 4 [10, 16, 19, 22, 23, 29–32, 59].

Compared with DARTS, our method with HB-inspired search space takes less search cost (0.3 GPU days vs. 0.4 GPU days), because HB-inspired search space contains fewer edges (see Figure 1). But still, our method results in a lower test error (2.82% vs. 3.00%). Furthermore, we use HB-inspired initialization and get a lower test error, 2.68%, which indicates that HBNet can be a good initial point to help search for a better architecture. We use the search space-AGD and get the cells shown in Figure 3. Compared with its counterpart DARTS, our searched architecture performs better (2.56% vs. 3.00%) using nearly the same search cost. We transfer the architecture searched in the search space-AGD from CIFAR-10 to ImageNet. As shown in Table 5 [5, 19, 23, 29, 30, 81–84], our method performs significantly better than DARTS (25.2% vs. 26.7% for top-1 error and 7.6% vs. 8.7% for top-5 error), which indicates the superiority of the search space-AGD.

Our results do not surpass P-DARTS [31], CNAS [59], and PC-DARTS [32] on ImageNet. Note that we just perform our method upon DARTS to test the effectiveness. Our method brings interpretability by offering optimization-inspired initial search point and search space, but does not change the search method, while P-DARTS, CNAS, and PC-DARTS propose better search methods to reduce the architectural gap,

Table 4 Experimental results on CIFAR-10 with our optimization-inspired initialization or search space and some competitive baselines in NAS, including DARTS[a)]

Methods	Test error (%)	Params (M)	Search cost (GPU days)	Search method
DenseNet-BC [10]	3.46	25.6	–	Manual
AmoebaNet-B +cutout [23]	2.55	2.8	3150	Evolution
ENAS + cutout [16]	2.89	4.6	0.5	RL
NASNet-A + cutout [19]	2.65	3.3	1800	RL
Hierarchical evolution [22]	3.75	15.7	300	Evolution
NASONet-WS [28]	3.53	3.1	0.4	NAO
DARTS (1st order) + cutout [29]	3.00	3.3	0.4	Gradient-based
DARTS (2nd order) + cutout [29]	2.76	3.3	1	Gradient-based
SNAS (moderate) + cutout [30]	2.85	2.8	1.5	Gradient-based
PC-DARTS + cutout [32]	2.57	3.6	0.1	Gradient-based
P-DARTS + cutout [31]	2.50	3.4	0.3	Gradient-based
CNAS + cutout [59]	2.60	3.7	0.3	Gradient-based
Search space-HB + cutout	2.82	2.92	0.3	Gradient-based
Search space-HB† + cutout	2.68	3.35	0.3	Gradient-based
Search space-AGD + cutout	2.56	3.3	0.4	Gradient-based

a) The mark † denotes using HBNet as initialization.

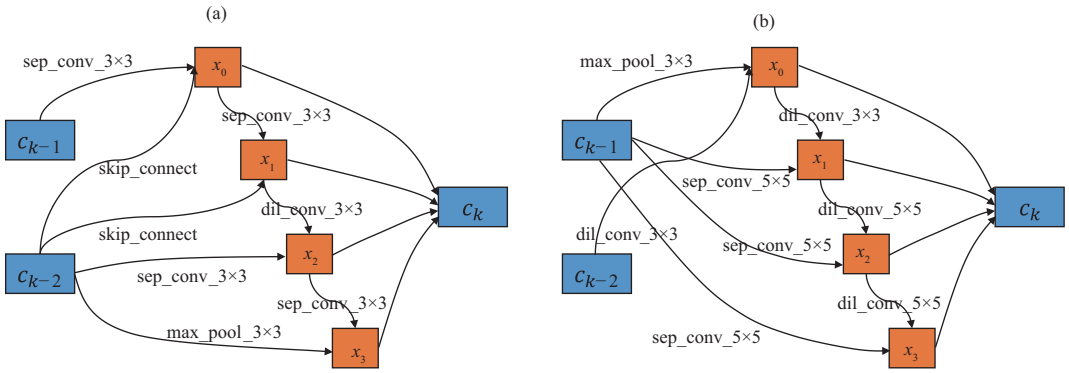

Figure 3 Our searched cells by search space-AGD on CIFAR 10. (a) Normal cell; (b) reduction cell.

Table 5 Comparison between our performance on ImageNet using the architecture searched by search space-AGD on CIFAR-10, and some competitive results in NAS, including DARTS, SNAS, and BayesNAS

Methods	Top-1 error (%)	Top-5 error (%)	Params (M)	Search method
MobileNet [81]	29.4	10.5	4.2	Manual
Inception-v1 [5]	30.2	10.1	6.6	Manual
ShuffleNet 2× [82]	25.1	–	∼5	Manual
AmoebaNet-C [23]	24.3	7.6	6.4	Evolution
MnasNet-92 [83]	25.2	8.0	5.5	RL
NASNet-A [19]	26.0	8.4	5.3	RL
DARTS (2nd order) [29]	26.7	8.7	4.7	Gradient-based
SNAS [30]	27.3	9.2	4.3	Gradient-based
BayesNAS [84]	26.5	8.9	3.9	Gradient-based
Ours (search space-AGD on CIFAR-10)	25.2	7.6	4.83	Gradient-based

alleviate the space explosion problem, and improve the search efficiency, respectively. So it is easy to integrate our proposed interpretable initial search point and search space onto these improved search methods. They are compatible and are expected to further improve the search results.

8 Conclusion and future work

In this study, we propose to leverage optimization algorithms to inspire manual architecture design and NAS. The conjecture that the optimization algorithm with a good convergence rate may imply a neural architecture with good performance is proposed and empirically validated. We prove that under certain conditions, the forward propagation in a vanilla DNN is equivalent to the iterative procedure of the GD algorithm to minimize a cost function. We derive good neural architectures, HBNet and AGDNet, by replacing the GD algorithm with more advanced optimization algorithms, the heavy ball algorithm and Nesterov's accelerated gradient algorithm. Surprisingly, the popular architectures, ResNet and DenseNet, can be regarded as the special cases of our HBNet and AGDNet, respectively.

As a limitation of our study, we have not proved the correspondence between optimization algorithm and neural architectures in a rigorous way. However, our optimization-inspired methodology can serve as a starting point to explain neural architectures. Our work allows practitioners to easily make changes, such as integrating engineering tricks, based on the optimization-inspired architectures for better results on manual architecture design and NAS. Such a practice should be much easier and more explainable, than designing manually from scratch or searching without any guidance.

Acknowledgements This work was supported by National Key R&D Program of China (Grant No. 2022ZD0160302) and National Natural Science Foundation of China (Grant No. 62276004).

Supporting information Appendixes A–D. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Krzhevsky A, Sutshever I, Hinton G. ImageNet classification with deep convolutional neural networks. In: Proceedings of Advances in Neural Information Processing Systems, 2012
2 Girshick R. Fast R-CNN. In: Proceedings of IEEE International Conference on Computer Vision, 2015
3 Ren S, He K, Girshick R, et al. Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell, 2016, 39: 1137–1149
4 Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2015
5 Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2015
6 Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In: Proceedings of International Conference on Learning Representations, 2015
7 Szegedy C, Vanhoucke V, Ioffe S, et al. Rethingking the inception architecture for computer vison. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2016
8 Srivastava RK, Greff K, Schmidhuber J. Training very deep networks. In: Proceedings of Advances in Neural Information Processing Systems, 2015. 2377–2385
9 He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2015
10 Huang G, Liu Z, van der Maaten L, et al. Densely connected convolutional networks. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2017
11 Chen Y, Li J, Xiao H, et al. Dual path networks. In: Proceedings of Advances in Neural Information Processing Systems, 2017
12 Yang Y, Zhong Z, Shen T, et al. Convolutional neural networks with alternately updated clique. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2018
13 Baker B, Gupta O, Naik N, et al. Designing neural network architectures using reinforcemen learning. In: Proceedings of International Conference on Learning Representations, 2017
14 Zoph B, Le Q. Neural architecture search with reinforcement learning. In: Proceedings of International Conference on Learning Representations, 2017
15 Liu C, Zoph B, Neumann M, et al. Progressive neural architecture search. In: Proceedings of European Conference on Computer Vision, 2018
16 Pham H, Guan M Y, Zoph B, et al. Efficient neural architecture search via parameter sharing. In: Proceedings of International Conference on Machine Learning, 2018
17 Zhong Z, Yan J, Wu W, et al. Practical block-wise neural network architecture generation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2018
18 Zhong Z, Yang Z, Deng B, et al. BlockQNN: efficient block-wise neural network architecture generation. IEEE Trans Pattern Anal Mach Intell, 2021, 43: 2314–2328
19 Zoph B, Vasudevan V, Shlens J, et al. Learning transferable architectures for scalable image recognition. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2018
20 Cai H, Yang J, Zhang W, et al. Path-level network transformation for efficient architecture search. In: Proceedings of International Conference on Machine Learning, 2018. 678–687
21 Real E, Moore S, Selle A, et al. Large-scale evolution of image classifiers. In: Proceedings of International Conference on Machine Learning, 2017. 2902–2911
22 Liu H, Simonyan K, Vinyals O, et al. Hierarchical representations for efficient architecture search. In: Proceedings of International Conference on Learning Representations, 2018

23 Real E, Aggarwal A, Huang Y, et al. Regularized evolution for image classifier architecture search. In: Proceedings of AAAI Conference on Artificial Intelligence, 2019. 4780–4789
24 Elsken T, Metzen J H, Hutter F. Efficient multi-objective neural architecture search via lamarckian evolution. In: Proceedings of International Conference on Learning Representations, 2019
25 Lu Z, Whalen I, Boddeti V, et al. NSGA-Net: neural architecture search using multi-objective genetic algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference, 2019. 419–427
26 Lu Z, Deb K, Goodman E, et al. NSGANetV2: evolutionary multi-objective surrogate-assisted neural architecture search. In: Proceedings of European Conference on Computer Vision, 2020. 35–51
27 Fang J, Chen Y, Zhang X, et al. EAT-NAS: elastic architecture transfer for accelerating large-scale neural architecture search. Sci China Inf Sci, 2021, 64: 192106
28 Luo R, Tian F, Qin T, et al. Neural architecture optimization. In: Proceedings of Advances in Neural Information Processing Systems, 2018. 7816–7827
29 Liu H, Simonyan K, Yang Y. Darts: differentiable architecture search. In: Proceedings of International Conference on Learning Representations, 2019
30 Xie S, Zheng H, Liu C, et al. SNAS: stochastic neural architecture search. In: Proceedings of International Conference on Learning Representations, 2019
31 Chen X, Xie L, Wu J, et al. Progressive differentiable architecture search: bridging the depth gap between search and evaluation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019. 1294–1303
32 Xu Y, Xie L, Zhang X, et al. PC-DARTS: partial channel connections for memory-efficient differentiable architecture search. In: Proceedings of International Conference on Learning Representations, 2020
33 Fang J, Sun Y, Zhang Q, et al. Densely connected search space for more flexible neural architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020. 10628–10637
34 Wu B, Dai X, Zhang P, et al. FBNet: hardware-aware efficient ConvNet design via differentiable neural architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019. 10734–10742
35 Weinan E. A proposal on machine learning via dynamical systems. Commun Math Stat, 2017, 5: 1–11
36 Haber E, Ruthotto L. Stable architectures for deep neural networks. Inverse Problems, 2017, 34: 014004
37 Chen T Q, Rubanova Y, Bettencourt J, et al. Neural ordinary differential equations. In: Proceedings of Advances in Neural Information Processing Systems, 2018. 6571–6583
38 Yang Y, Wu J, Li H, et al. Dynamical system inspired adaptive time stepping controller for residual network families. In: Proceedings of AAAI Conference on Artificial Intelligence, 2020
39 Gregor K, LeCun Y. Learning fast approximations of sparse coding. In: Proceedings of International Conference on Machine Learning, 2010
40 Xin B, Wang Y, Gao W, et al. Maximal sparsity with deep networks. In: Proceedings of Advances in Neural Information Processing Systems, 2016
41 Kulkarni K, Lohit S, Turaga P, et al. ReconNet: non-iterative reconstruction of images from compressively sensed mmeasuremets. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2016
42 Zhang J, Ghanem B. ISTA-Net: iterative shrinkage-thresholding algorithm inspired deep network for image compressie sensing. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2018
43 Yang Y, Sun J, Li H, et al. Deep ADMM-Net for compressive sensing MRI. In: Proceedings of Advances in Neural Information Processing Systems, 2016
44 Xie X, Wu J, Zhong Z, et al. Differentiable linearized ADMM. In: Proceedings of International Conference on Machine Learning, 2019
45 Schaffer J, Whitley D, Eshelman L. Combinations of genetic algorithms and neural networks: a survey of the state of the art. In: Proceedings of International Workshop on Combinations of Genetic Algorithms and Neural Networks, 1992
46 Leung F H F, Lam H K, Ling S H, et al. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. IEEE Trans Neural Netw, 2003, 14: 79–88
47 Verbancsics P, Harguess J. Generative neuroevolution for deep learning. 2013. ArXiv:1312.5355
48 Saxena S, Verbeek J. Convolutional neural fabrics. In: Proceedings of Advances in Neural Information Processing Systems, 2016
49 Domhan T, Springenberg J, Hutter F. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In: Proceedings of International Joint Conference on Artificial Intelligence, 2015
50 Bergstra J, Yamins D, Cox D. Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. In: Proceedings of International Conference on Machine Learning, 2013
51 Kwok T, Yeung D. Constructive algorithms for structure learning in feedforward neural networks for regression problems. IEEE Trans Neural Netw, 1997, 8: 630–645
52 Ma L, Khorasani K. A new strategy for adaptively constructing multilayer feedforward neural networks. Neurocomputing, 2003, 51: 361–385
53 Cortes C, Gonzalvo X, Kuznetsov V, et al. AdaNet: adaptive structure learning of artificial nerual networks. In: Proceedings of International Conference on Machine Learning, 2017
54 Brock A, Lim T, Ritchie J M, et al. SMASH: one-shot model architecture search through hypernetworks. In: Proceedings of International Conference on Learning Representations, 2018
55 Cai H, Zhu L, Han S. ProxylessNAS: direct neural architecture search on target task and hardware. In: Proceedings of International Conference on Learning Representations, 2019
56 Bender G, Kindermans P J, Zoph B, et al. Understanding and simplifying one-shot architecture search. In: Proceedings of International Conference on Machine Learning, 2018. 550–559
57 Guo Z, Zhang X, Mu H, et al. Single path one-shot neural architecture search with uniform sampling. In: Proceedings of European Conference on Computer Vision, 2020. 544–560
58 Stamoulis D, Ding R, Wang D, et al. Single-path NAS: designing hardware-efficient convnets in less than 4 hours. In: Proceedings of Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2019. 481–497
59 Guo Y, Chen Y, Zheng Y, et al. Breaking the curse of space explosion: towards efficient nas with curriculum search. In: Proceedings of International Conference on Machine Learning, 2020. 3822–3831
60 Yang Y, You S, Li H, et al. Towards improving the consistency, efficiency, and flexibility of differentiable neural architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021. 6667–6676
61 Yang Y, Li H, You S, et al. ISTA-NAS: efficient and consistent neural architecture search by sparse coding. In: Proceedings

of Advances in Neural Information Processing Systems, 2020. 10503–10513
62 Beck A, Teboulle M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J Imag Sci, 2009, 2: 183–202
63 Sprechmann P, Bronstein A M, Sapiro G. Learning efficient sparse and low rank models. IEEE Trans Pattern Anal Mach Intell, 2015, 37: 1821–1833
64 Zhou J T, Di K, Du J, et al. SC2Net: sparse LSTMs for sparse coding. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, 2018
65 Chen X, Liu J, Wang Z, et al. Theoretical linear convergence of unfolded ISTA and its practical weights and thresholds. In: Proceedings of Advances in Neural Information Processing Systems, 2018. 9061–9071
66 Metzler C, Mousavi A, Baraniuk R. Learned D-AMP: principled neural network based compressive image recovery. In: Proceedings of Advances in Neural Information Processing Systems, 2017. 1772–1783
67 Li H, Yang Y, Chen D, et al. Optimization algorithm inspired deep neural network structure design. In: Proceedings of Asian Conference on Machine Learning, 2018. 614–629
68 Bertsekas D. Nonlinear Programming. Belmont: Athena Scientific, 1999
69 Polyak B T. Some methods of speeding up the convergence of iteration methods. USSR Comput Math Math Phys, 1964, 4: 1–17
70 Nesterov Y. A method for unconstrained convex minimization problem with the rate of convergence $O(1 = k^2)$. Sov Math Dokl, 1983, 27: 372–376
71 Gabay D. Applications of the method of multipliers to variational inequalities. Stud Math Appl, 1983, 15: 299–331
72 Lin Z, Liu R, Su Z. Linearized alternating direction method with adaptive penalty for low-rank representation. In: Proceedings of Advances in Neural Information Processing Systems, 2011
73 Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of Artificial Intelligence and Statistics, 2010
74 He K, Zhang X, Ren S, et al. Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of IEEE International Conference on Computer Vision, 2015
75 Lin M, Chen Q, Yan S. Network in network. In: Proceedings of International Conference on Learning Representations, 2014
76 Springenberg J T, Dosovitskiy A, Brox T, et al. Striving for simplicity: the all convolutional net. In: Proceedings of International Conference on Learning Representations Workshop, 2015
77 Lee C Y, Xie S, Gallagher P, et al. Deeplysupervised nets. In: Proceedings of Artificial Intelligence and Statistics, 2015. 562–570
78 Loshchilov I, Hutter F. SGDR: stochastic gradient descent with warm restarts. In: Proceedings of International Conference on Learning Representations, 2017
79 Kingma D P, Ba J. ADAM: a method for stochastic optimization. In: Proceedings of International Conference on Learning Representations, 2015
80 DeVries T, Taylor G W. Improved regularization of convolutional neural networks with cutout. 2017. ArXiv:1708.04552
81 Howard A G, Zhu M, Chen B, et al. MobileNets: efficient convolutional neural networks for mobile vision applications. 2017. ArXiv:1704.04861
82 Ma N, Zhang X, Zheng H T, et al. ShuffleNet V2: practical guidelines for efficient cnn architecture design. In: Proceedings of European Conference on Computer Vision, 2018. 116–131
83 Tan M, Chen B, Pang R, et al. MnasNet: platform-aware neural architecture search for mobile. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019. 2820–2828
84 Zhou H, Yang M, Wang J, et al. BayesNAS: a Bayesian approach for neural architecture search. In: Proceedings of International Conference on Machine Learning, 2019. 7603–7613