Cyclic Differentiable Architecture Search

Hongyuan Yu, Houwen Peng, Yan Huang, Jianlong Fu, Hao Du, Liang Wang, and Haibin Ling

Abstract—Differentiable ARchiTecture Search, i.e., DARTS, has drawn great attention in neural architecture search. It tries to find the optimal architecture in a shallow search network and then measures its performance in a deep evaluation network. The independent optimization of the search and evaluation networks, however, leaves a room for potential improvement by allowing interaction between the two networks. To address the problematic optimization issue, we propose new joint optimization objectives and a novel Cyclic Differentiable ARchiTecture Search framework, dubbed CDARTS. Considering the structure difference, CDARTS builds a cyclic feedback mechanism between the search and evaluation networks with introspective distillation. First, the search network generates an initial architecture for evaluation, and the weights of the evaluation network are optimized. Second, the architecture weights in the search network are further optimized by the label supervision in classification, as well as the regularization from the evaluation network through feature distillation. Repeating the above cycle results in a joint optimization of the search and evaluation networks and thus enables the evolution of the architecture to fit the final evaluation network. The experiments and analysis on CIFAR, ImageNet and NATS-Bench [95] demonstrate the effectiveness of the proposed approach over the state-of-the-art ones. Specifically, in the DARTS search space, we achieve 97.52% top-1 accuracy on CIFAR10 and 76.3% top-1 accuracy on ImageNet. In the chain-structured search space, we achieve 78.2% top-1 accuracy on ImageNet, which is 1.1% higher than EfficientNet-B0. Our code and models are publicly available at https://github.com/microsoft/Cream.

Index Terms—Cyclic, introspective distillation, differentiable architecture search, unified framework

1 Introduction

Deep learning has enabled remarkable progress in a variety of vision tasks over the past years. One crucial factor for this progress is the design of novel neural network architectures. Most of current employed architectures are designed by human experts, which is time-consuming and error-prone. Because of this, there is a growing interest in automatic Neural Architecture Search (NAS) for vision tasks, such as image recognition [1], [2], object detection [3], [4] and semantic segmentation [5], [6].

Differentiable architecture search, i.e., DARTS [7], has become one of the most popular NAS pipelines nowadays due to its relatively low computational cost and competitive performance [8]. Different from previous methods [1], [9], [10], [11] that search over a discrete set of candidate architectures, DARTS relaxes the search space to be continuous, so that the architecture can be optimized by the gradient descent. The efficiency of gradient-based optimization reduces the search cost from thousands of GPU days to a few. According to the recent NAS survey [8], [12], [13], due to the simplicity and elegance of the DARTS architecture, the research work related to DARTS is quite rich [14], [15], [16]. Moreover, gradient optimization in a continuous search strategy is an important trend of NAS.

Existing DARTS approaches have to divide the search process into two steps: search and evaluation, as shown in Fig. 1a. The search step employs a small network to discover the optimal cell[1] structures, and the evaluation step stacks the discovered cells to construct a large network for final evaluations. This results in the optimization of search process is independent from the target evaluation network. As shown in Fig. 1b, PDARTS [15] tried to alleviate the gap by gradually deepening the search network. In Fig. 1c, EnTranNAS [16] built the search network by combining the evaluation network module and the search network module to reduce the gap. Besides, SNAS [17] and GDAS [18] applied Gumbel-softmax and modified straight-through Gumbel-Softmax to relieve the gap caused by discretization. Furthermore, AutoHAS [19] augmented GDAS with an entropy term to search for both hyperparameters and architectures. However, those methods still separates the search and evaluation process. As a consequence, the performance of discovered architectures in the search network has limited correlation with the actual performance of the evaluation

- *Hongyuan Yu, Yan Huang, and Liang Wang are with the Center for Research on Intelligent Perception and Computing (CRIPAC), National Laboratory of Pattern Recognition (NLPR), Center for Excellence in Brain Science and Intelligence Technology (CEBSIT), AI School, University of Chinese Academy of Sciences (UCAS), Beijing 101408, China. E-mail: hongyuan.yu@cripac.ia.ac.cn, {yhuang, wangliang}@nlpr.ia.ac.cn.*
- *Houwen Peng and Jianlong Fu are with Microsoft Research, Beijing 100080, China. E-mail: {houwen.peng, jianf}@microsoft.com.*
- *Hao Du is with the Department of Computer Science, City University of Hong Kong, Hong Kong, China. E-mail: haodu8-c@my.cityu.edu.hk.*
- *Haibin Ling is with the Department of Computer Science, Stony Brook University, Stony Brook, NY 11794 USA. E-mail: haibin.ling@stonybrook.edu.*

Manuscript received 9 November 2020; revised 28 January 2022; accepted 9 February 2022. Date of publication 23 February 2022; date of current version 5 December 2022.
This work was supported in part by the National Key Research and Development Program of China under Grant 2018AAA0100400, in part by the National Natural Science Foundation of China under Grants 61721004, U1803261, and 61976132, in part by Beijing Nova Program under Grant Z201100006820079, in part by the Key Research Program of Frontier Sciences CAS under Grant ZDBS-LY-JSC032, and in part by CAS-AIR.
(Corresponding author: Houwen Peng.)
Recommended for acceptance by Z. Tu.
Digital Object Identifier no. 10.1109/TPAMI.2022.3153065

1. Cell is a basic building block for network construction. It consists of convolution, pooling, nonlinearity and a prudent selection of connections.

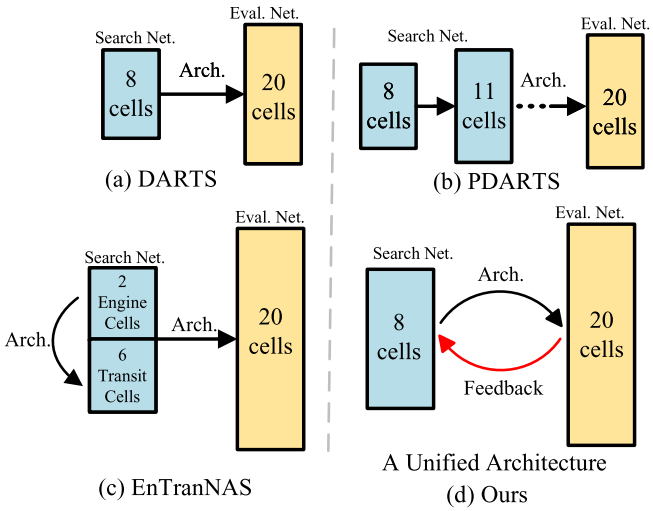


Fig. 1. Comparisons of prior DARTS and our CDARTS. In prior DARTS [7], PDARTS [15], and EnTranNAS [16], the target evaluation network does not engage in the architecture search. In contrast, our CDARTS combines the search and evaluation networks into a joint optimization framework. The **blue** and **gold** boxes indicate the search and evaluation networks, respectively.

network. On the other hand, there are several recent works casting doubt on the effectiveness of DARTS. Li and Talwalkar [20] observe that even a simple random search method can find architectures outperforming the original DARTS. Zela *et al.* [21] and Liang *et al.* [22] show that DARTS is prone to degenerate to networks filled with parametric-free operations, e.g., skip connections, leading to a poor performance of the searched architecture. Wang *et al.* [23] find that the problematic optimization of DARTS results in the learned architecture weights are insufficient to reflect the relative ranking of different architectures during evaluation [16], [24], [25], [26].

To alleviate these issues, we propose a *cyclic differentiable architecture search* method, dubbed CDARTS. CDARTS integrates the search and evaluation networks into a unified architecture, and jointly trains the two networks in a cyclic way. As visualized in Fig. 1c, the shallow search network provides the best intermediate architecture to the evaluation network. In turn, the search network gets the feedback from the evaluation network (with higher model capacity due to more layers) by introspective distillation. The distillation is introspective because it does not require introducing third-party models, such as human-designed architectures, to serve as the teacher models. This is crucial for practical applications because there is usually no available teacher model in new tasks. As proved by Liu *et al.* [27] and Li *et al.* [28], the knowledge of a network model not only lies in the network parameters but also the network structure. The introspective distillation could transfer the knowledge embedded in parameters as well as the the knowledge inside the structure from the evaluation network to the search network.

Moreover, instead of training the search and evaluation networks separately, we propose a joint learning algorithm to optimize the integrated architecture in a cyclic manner. It consists of a separate learning and a joint learning stage. The separate learning stage aims to optimize the search and evaluation networks to have good initializations. The joint learning stage is to update the architecture and network weights alternatively. Specifically, the joint learning stage first optimizes the architecture weights and the evaluation network weights by jointly training the search and evaluation networks. Then, it optimizes the search network weights according to the updated architecture weights. These two learning stages are performed alternatively, leading to a cyclic optimization between the search and evaluation networks. Eventually, the target evaluation network obtains a shaped architecture tailored by the search network. The proposed jointly training scheme allows the direct optimization of the evaluation network, reliving the problematic optimization of DARTS [23].

We evaluate our CDARTS algorithm on image classification task and conduct experiments on CIFAR [29], ImageNet [30], as well as the recently proposed NATS-Bench [95] benchmark [31]. The experiments demonstrate that, on CIFAR, CDARTS achieves superior performance to existing state-of-the-art DARTS approaches. On the large-scale ImageNet, the proposed CDARTS also shows its superiority in the DARTS family, while achieving comparable performance to MobileNet-V3 [32], which blends automatic search techniques with human intuition. Moreover, for a fair comparison with other NAS algorithms, such as one-shot [18], [33] and reinforcement learning-based [34], [35] approaches, we conduct an experiment on the NATS-Bench [95] benchmark, which provides a fixed search space with a unified training setting. The experiment shows that CDARTS achieves competitive performance compared with 10 recent prevailing NAS approaches. Last but not the least, we also apply our introspective distillation search method to the chain-structured search space [12] and achieve a gain of 1.6% over EfficientNet-B0 [36] on ImageNet with a similar model size. We further transfer the discovered architectures to the downstream object detection task and semantic segmentation task. For object detection, we get an AP of 36.2 on the COCO validation set, which surpasses the state-of-the-art MobileNetV3 by 6.3%. And for semantic segmentation, we obtain an mIoU of 78.2% on the cityscapes validation set and 40.4% on ADE20K validation set, which are 1.9% and 3% superior than the recent SegFormer [37], respectively.

Our main contributions are summarized as below:

- We rethink the bi-level optimization for NAS and empirically show that the performance of discovered architectures in the search network has a limited correlation with the actual performance of the evaluation network, leading to the instability issue. We propose brand-new joint optimization objectives for differentiable one-shot NAS.
- We integrate the search and evaluation networks into a unified framework, and propose a cyclic learning algorithm. Such algorithm addresses the problematic optimization issue in previous differentiable methods [7], [15], [38], thus being able to improve the stability of the search algorithm.
- We propose an introspective distillation mechanism to transfer the knowledge embedded in both parameters and structure from the evaluation network to the search network. In contrast to classical distillation methods [28], [39], such introspective method

does not require a predefined third-party teacher model, which is more flexible in practice.
- Extensive experiments on three types of search spaces verify the robustness of the proposed method. Moreover, the results of the searched architectures in the object detection and semantic segmentation tasks demonstrate their generality and transferability.

2 Related Work

Deep neural networks (DNN) have played an important role in computer vision tasks such as image recognition [30], [40], [41], object detection [42], [43], segmentation [44], [45], [46], tracking [47], and so on [48], [49], [50]. However, designing effective and efficient neural architectures is a labor-intensive process that may require lots of trials by human experts. Hence, automatic neural architecture search (NAS) [51], [52], [53], [54] has emerged in recent years. In this section, we first summarize the search space, and then discuss the search strategies in recent NAS methods. At last, we briefly review some distillation methods which are most related to our method.

2.1 Search Space

The search space defines all possible architectures in principle. A well designed search space can simplify the search. Current search spaces mainly fall into two categories: chain-structured space and cell-based space.

In the chain-structured search space, the neural architecture can be represented by a sequence of stacked layers. Baker *et al.* [55] consider a set of layers that includes convolutions, pooling and dense connections. The corresponding hyperparameter settings contain the number of filters, kernel size, stride and pooling size. Zoph *et al.* [9] define a relaxed version of the chain-structured search space by introducing skip connections between blocks. Xie *et al.* [56] extend the search space by replacing feature concatenation operations with summations when merging features. Most recently, the inverted residual blocks are introduced to facilitate the search of neural architectures for mobile platforms. MNAS [57], ProxylessNAS [14] and other related methods [28], [58], [59] searches architectures over the space consisting of the lightweight inverted bottleneck MBConvs [60]. MobileNetV3 [32] and OFA [61] extend the space with the Swish activation function and squeeze-excitation modules, thus achieving impressive results under mobile settings.

Although models searched based on chain-structured search space have shown strong representation ability, it generally consumes extensive computation cost to exhaustively cover the search space during training. Motivated by hand-crafted architectures consisting of repeated motifs [62], [63], [64], Zoph *et al.* [1] and Zhong *et al.* [65] first build networks by repeating the searched cells which is easy to scale the model size by adjusting the number of cell [66].

The development of NAS algorithms in different kinds of search spaces indicates that a better search space is essential for designing search strategies. Recently, some works [21], [25] reveal that even random sampled cells can get pretty good results under those carefully designed search space. Therefore, several NAS benchmarks [31], [67], [68] are released to eliminate the bias of search space and relieve the evaluation burden. In this paper, we apply our method to both chain-structured search space and cell-base search space and achieve promising results on CIFAR [29] and ImageNet [30]. Furthermore, we verify the efficiency of our method on NATS-Bench [31], [95] .

2.2 Search Strategy

Early NAS approaches focus on searching a network motif by using either reinforcement learning [1], [9] or evolution algorithms [56], [66], [69], [70], [71], [72] and build a complete network for evaluation by stacking the motif. Unfortunately, these approaches require to train hundreds or thousands of candidate architectures from scratch, resulting in barely affordable computational overhead. Thus, several follow-up works are proposed to speed-up training by reducing the search space [10], [73].

More recent works resort to the weight-sharing *one-shot* model to amortize the searching cost [20], [35], [54], [74]. The key idea of one-shot approach is to train a single over-parameterized model, and then share the weights between sampled child models. This weight sharing mechanism allows the searching of a high-quality architecture within a few GPU days [75], [76]. Single path one-shot model families [20], [28], [61], [77], [78] propose to train a single sampled path of the one-shot model in each iteration, rather than the entire over-parameterized model. Once the training process is finished, the child models can be ranked by the shared weights. However, [21] finds that the weight sharing strategy results in inaccurate performance evaluation of the candidate architecture, making it difficult for the one-shot NAS to identify the best architecture. FairNAS [79] and PC-NAS [80] also demonstrate that neural architecture candidates based on these parameter sharing methods also cannot be adequately trained, which leads to an inaccurate ranking of architecture candidates . There are few works leveraging knowledge distillation [39] to boost the training of the super network, and they commonly introduce additional large models as teachers. More specifically, OFA [61] pretrains the largest model in the search space and use it to guide the training of other subnetworks, while DNA [28] directly employs the third-party EfficientNet-B7 [36] as the teacher model. These search algorithms will become infeasible if there is no available pretrained model, especially when the search task and data are entirely new. In contrast to those methods, our introspective distillation method does not require a predefined third-party teacher model, which is more flexible in practice.

Differentiable architecture search, i.e., DARTS [7], is another representative one-shot model. Instead of searching over a discrete set of candidate architectures, DARTS relaxes the search space to be continuous, so that the architecture can be optimized by the efficient gradient descent. Despite DARTS has achieved promising performance by only using orders of magnitude less computation resources, some issues remain. ProxylessNAS [14] argues that the optimization objectives of search and evaluation networks are inconsistent in DARTS [7]. Thus it employs a binary connection to tackle the issue. PDARTS [15] points out the depth gap problem of DARTS, and thereby presents a progressive learning approach, which gradually increases the number of layers in the search network. Moreover, PCDARTS [81]

addresses the problem of high GPU memory cost through introducing a partially-connected strategy in network optimization. SNAS [17] and GDAS [18] tried to relieve the discretization gap with modified Gumbel-Softmax. Furthermore, AutoHAS [19] augmented GDAS with an entropy term to search for both hyperparameters and architectures. ISTA-NAS [82] formulated neural architecture search as a sparse coding problem to avoid the post-processing of DARTS. Moreover, EnTranNAS [16] proposed an architecture derivation method to replace the hand-crafted post-processing rules.

Our proposed approach differs from existing DARTS approaches [7], [15], [38], [81] in two major ways. One is that our approach integrates the search and evaluation networks into a unified architecture. Also, we build up connections to transfer the parameters as well as the structure knowledge of the evaluation network to the search network. The other difference is the searching algorithm. In contrast to previous methods where the optimization of architecture weights is independent from the evaluation network, our approach enables the evaluation network to guide the search of architectures by joint training.

2.3 Knowledge Transfer Between Architectures

Transferring knowledge between architectures during the search is widely used in NAS. A simple way is sharing weights across architectures. Such way can largely reduce training cost because the knowledge from previous searches is reused. TAPAS [83] starts with a simple architecture and scales up it based on a prediction model. T-NAML [84] adapts a pre-trained network to target datasets with reinforcement learning, which makes decisions on optimizing neural architectures across datasets simultaneously. XferNet [85] accelerates NAS by transferring architecture knowledge across different tasks. NATS [86] introduces a practical neural architecture transformation search algorithm for object detection. It explores architecture space based on existing networks and reusing their weights without searching and re-training. Instead of only inheriting weights from previously-trained networks, FNA++ [87] adapt both the architecture and parameters of a seed network, which makes it possible to use NAS for segmentation and detection more efficiently. NAT [88] is also designed to efficiently generate task-specific models under multiple searching objectives.

Knowledge distillation [39] is another widely used technique for knowledge transfer between architectures. It compresses the "dark knowledge" of a well trained larger model to a smaller one [89]. An enormous amount of approaches have been proposed to fortify the efficiency of student models' learning capability. Recently, one-shot methods try to improve the search process by using the supervision of other high-capacity networks, and they commonly introduce additional large models as teachers. BIGNAS [58] and OFA [61] pretrain the largest model in the search space and use it to guide the training of other subnetworks, while DNA [28] directly employs the third-party EfficientNet-B7 [36] as the teacher model. These search algorithms will become infeasible if there is no available pre-trained model, especially when the search task and data are entirely new. Different from these methods, the teacher network of our method is generated by the model itself. Hence, our model does not need to spend extra time to train a teacher network alone and thereby is more flexible.

3 Methodology

In this section, we present the Cyclic Differentiable Architecture Search method, i.e., CDARTS. We first briefly review the standard gradient-based differentiable method, which trains the search and evaluation networks independently [7], [15], [38], [81]. Then, we propose the cyclic search method, which learns the search and evaluation networks jointly. At the end, we present the implementation details of the network architecture in our method.

3.1 Preliminary: Differentiable Architecture Search

The goal of Differentiable Architecture Search, i.e., DARTS, [7] is to search a cell motif, which can be stacked repeatedly to construct a convolutional network. A cell is a directed acyclic graph consisting of an ordered sequence of N nodes $\{x_i\}_{i=0}^{N-1}$. Each node x_i is a latent representation (*e.g.*, a feature map), while each directed edge (i,j) is associated with one operation $o^{(i,j)}$ which transforms information from x_i to x_j. Let $\mathcal{O}$ denote the operation space, consisting of a set of candidate operations, such as convolution, max-pooling, skip-connection, etc. Each operation represents a function $o(\cdot)$ to be performed on x_i. To make the search space continuous, DARTS relaxes the choice of a particular operation to a softmax over all possible operations [7]:

$$\bar{o}^{(i,j)}(x_i) = \sum_{o \in \mathcal{O}} \frac{\exp(\boldsymbol{\alpha}_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\boldsymbol{\alpha}_{o'}^{(i,j)})} \cdot o(x_i), \tag{1}$$

where the operation weights for a pair of nodes (i,j) are parameterized by the architecture weights $\boldsymbol{\alpha}(i,j)$ of dimension $|\mathcal{O}|$. Here, the parameter $\boldsymbol{\alpha}$ is the encoding of architectures to be optimized. An intermediate node of the cell is computed based on all of its predecessors as $x_j = \sum_{i<j} \bar{o}^{(i,j)}(x_i)$, and the output node x_{N-1} is obtained by concatenating all the intermediate nodes in the channel dimension.

With the definition of a cell, the search of the optimal architecture becomes to solve the following bilevel optimization problem:

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

where $\boldsymbol{\alpha}$ contains the architecture weights optimized on the validation data (*val*), and w_S denotes the parameters of the *search* network learnt on the training data (*train*). Eq. (2) amounts to optimize the network and architecture weights, i.e., w_S and $\boldsymbol{\alpha}$, in an alternative way. After getting the optimal architecture, DARTS constructs a new evaluation network by stacking the discovered neural cells and retrains from scratch over the training of the target task.

Without loss of generality, here we only consider one cell from a simplified search space consists of two operations: (skip, conv). According to Wang *et al.* [23], the current estimation of the optimal feature map m^*, which is shared across all edges, can then be written as:

$$\bar{m}_e(x_e) = \frac{\exp(\alpha_{conv})}{\exp(\alpha_{\text{conv}}) + \exp(\alpha_{skip})} o_e(x_e) + \frac{\exp(\alpha_{skip})}{\exp(\alpha_{\text{conv}}) + \exp(\alpha_{skip})} x_e \quad (3)$$

where α_{conv} and α_{skip} are the architecture parameters defined in DARTS. $o_e(x_e)$ is the output of the convolution operation, and x_e is the skip connection (i.e., the input feature map of edge e). By minimizing $\text{var}(\bar{m}_e - m^*)$, we can get the optimal α^*_{conv} and α^*_{skip} as:

$$\begin{aligned} \alpha^*_{conv} &\propto \text{var}(x_e - m^*), \\ \alpha^*_{skip} &\propto \text{var}(o_e(x_e) - m^*). \end{aligned} \quad (4)$$

During the training of the search network, x_e will be theoretically closer to m^* than $o_e(x_e)$, resulting in α_{skip} to be greater than α_{conv}. This leads to failed search because cells are full of skip-connections. Therefore, the magnitude of architecture parameters, essentially, cannot indicate how much the operation contributes to the search network's performance.

The above procedure shows that, in DARTS, the optimization of the evaluation network is separated from the search process of architectures, i.e., Eq. (2). As a consequence, the magnitude of architecture parameters cannot indicate how much the operation contributes to the search network's performance, leading to the discovered architectures are sub-optimal for the final evaluation networks.

3.2 Cyclic Differentiable Architecture Search

Different from previous methods [7], [15], [81] where the search network is separated from the evaluation network, our CDARTS integrates the two networks into a unified architecture and models the architecture search as a joint optimization problem of the two networks:

$$\min_{\alpha} \quad \mathcal{L}_{val}(w^*_E, w^*_S, \alpha), \quad \text{s.t.} \begin{cases} w^*_S = \arg\min_{w_S} \mathcal{L}_{train}(w_S, \alpha), \\ w^*_E = \arg\min_{w_E} \mathcal{L}_{val}(w_E, \alpha), \end{cases} \quad (5)$$

where w_E is the weight of the evaluation network. To optimize this objective function, we propose an alternating learning algorithm, consisting of two iterative stages: a *separate learning* stage and a *joint learning* stage. The former is to train the search and evaluation networks on the *train* and *val* datasets respectively, and enable them to have good initialization weights w_S and w_E . The latter is to learn the architecture weights α and the network weights jointly. These two stages are performed alternatively until convergence, leading to a joint optimization between the network search and evaluation, as presented in Algorithm 1. This cyclic process allows the evolution of searched architectures to fit the final target evaluation network.

Stage 1: Separate Learning. The goal of this stage is to train the search and evaluation networks individually and make them adaptive to the given data. Specifically, for the search network, the architecture weights α are initialized randomly before training. Then, the weights w_S are optimized on the $train$ data as follows:

$$w^*_S = \arg\min_{w_S} \mathcal{L}^S_{train}(w_S, \alpha), \quad (6)$$

where $\mathcal{L}^S_{train}$ defines the loss function, and w^*_S denotes the learned weight. For the image classification problem, we specify $\mathcal{L}^S_{train}$ as a cross-entropy loss.

For the evaluation network, its internal cell structures are generated by discretizing the learned weights α. Following the previous work [1], [7], we retain the top-k ($k = 2$) operations for each node in the cells by thresholding the learned values in α. The separate learning of the evaluation network is performed on the *val* set through optimizing the following objective function:

$$w^*_E = \arg\min_{w_E} \mathcal{L}^E_{val}(w_E, \overline{\alpha}), \quad (7)$$

where $\overline{\alpha}$ indicates the top-k discretization of the continuous α, and w^*_S represents the learned weight. The separate learning of w_S and w_E enables the search and evaluation networks to obtain a good initialization.

Algorithm 1. Cyclic Differentiable Architecture Search

Input: The *train* and *val* data, search and evaluation iterations S_S and S_E, update iterations S_U, architecture weights α, and weights w_S and w_E for search S-Net and evaluation E-Net.
Output: Evaluation network.

```
Initialize α with randoms
Initialize w_S
for each search step i ∈ [0, S_S] do
  /* Stage 1: Separate training */
  if i mod S_U = 0 then
    Discretize α to ᾱ by selecting the top-k elements
    Generate E-Net with ᾱ
    for each evaluation step j ∈ [0, S_E] do
      Calculate L^E_val according to Eq. (7)
      Update w_E
    end for
  end if
  /* Stage 2: Joint training */
  Calculate L^S_val, L^E_val and L^{S,E}_val according to Eq. (8)
  Jointly update α and w_E
  Calculate L^S_train according to Eq. (6)
  Update w_S
end for
```

Stage 2: Joint Learning. In this optimization stage, the search algorithm updates the architecture weights α with the feature feedback from the evaluation network through introspective distillation. More concretely, the joint optimization of the two networks is formulated as:

$$\alpha^*, w^*_E = \arg\min_{\alpha, w_E} \quad \mathcal{L}^S_{val}(w^*_S, \alpha) + \mathcal{L}^E_{val}(w_E, \overline{\alpha}) + \lambda \, \mathcal{L}^{S,E}_{val}(w^*_S, \alpha, w_E, \overline{\alpha}), \quad (8)$$

where minimizing $\mathcal{L}^S_{val}(w^*_S, \alpha)$ is to optimize the architecture weights α with the fixed weights w^*_S in the search network, $\mathcal{L}^E_{val}(w_E.\overline{\alpha})$ is to optimize the weights w_E with a fixed architecture $\overline{\alpha}$ in the evaluation network. The $\mathcal{L}^{S,E}_{val}(w^*_S, \alpha, w_E, \overline{\alpha})$ represents the proposed *introspective distillation*, which allows the knowledge transfer from the evaluation network to the search network. $\mathcal{L}^{S,E}_{val}(\cdot)$ employs the features derived

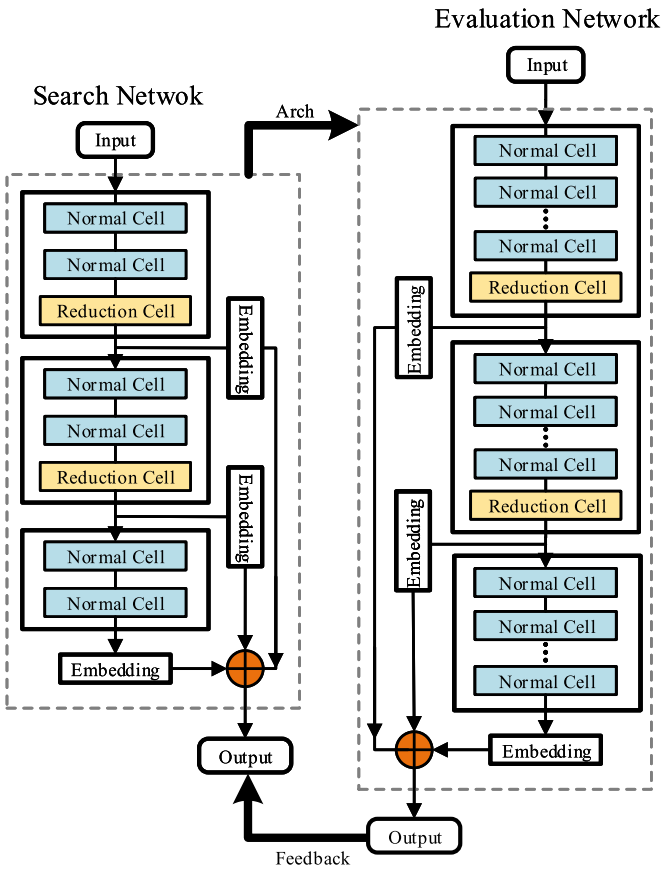


Fig. 2. Illustration of the proposed CDARTS. CDARTS contains two networks, the search network (left) and the evaluation network (right). The *Embedding* module maps each stage feature to a one-dimensional vector.

from the evaluation network as a supervision signal to guide the updates of the architecture hyperparameter α in the search network. The introspective distillation is formulated by a soft target cross entropy function as:

$$\mathcal{L}_{val}^{S,E}(w_S^*, \alpha, w_E, \overline{\alpha}) = \frac{T^2}{N}\sum_{i=1}^{N}\left(p(w_E, \overline{\alpha})\log\left(\frac{p(w_E, \overline{\alpha})}{q(w_S^*, \alpha)}\right)\right), \quad (9)$$

where N is the number of training samples, and T is a temperature coefficient (set to 2). Here, $p(\cdot)$ and $q(\cdot)$ represent the output feature logits of the evaluation and search networks respectively, each of which is calculated as the soft target distribution [39] over the feature logits, i.e.,

$$\begin{aligned} p(w_E, \overline{\alpha}) &= \frac{\exp(f_i^E/T)}{\sum_j \exp(f_j^E/T)}, \\ q(w_S^*, \alpha) &= \frac{\exp(f_i^S/T)}{\sum_j \exp(f_j^S/T)}, \end{aligned} \quad (10)$$

where f_i^E and f_i^S denote the features generated by the search and evaluation networks (see Fig. 2). The optimization of Eq. (9) distills the knowledge of the evaluation network to guide the updates of architecture weights in the search network, while the joint training in Eq. (8) allows the knowledge transfer between the two networks.

Compared to the classical distillation method [39], CDARTS performs knowledge transfer in an introspective manner such that the search network can get the informative feedback from the evaluation network consecutively. The introspective distillation has three advantages. First, the teacher model is endogenous, which enables our method to adapt to any scenario. For example, there is no need to train an extra teacher, which greatly saves computing resources. Second, instead of learning from a fixed teacher model, our method updates the teacher models during the search process, thus the teacher model can adapt to the learning state of the student model. This has also been proved by Pham *et al*. [90] that when the teacher and student network are alternately optimized, the student network may learn from better distributions and achieve good validation performance. Last but not the least, the introspective distillation enables the search network to get the feedback information from the current search status, which prevents the search process from collapse [22] and improves the search stability.

In addition, it has been observed that DARTS approaches tend to search the architectures with plenty of skip-connection operations rather than convolutions or poolings, because that the skip-connection allows rapid gradient descent [15], [22]. This is essentially a kind of overfitting of architecture search. To address this issue, we impose an ℓ_1-norm regularization on the weight of the skip-connection operation in the architecture weights α as:

$$\mathcal{L}_{reg} = \lambda' \|\alpha\|_1, \quad (11)$$

where $\|\cdot\|_1$ represents the ℓ_1 norm, and λ' is a positive tradeoff coefficient. Eq. (11) is finally optimized with Eq. (8) jointly as an auxiliary item to inhibit overfitting.

It is worth noting that during the separate learning of the evaluation network, i.e., Eq. (7), we use a weight-sharing strategy for updating weight w_E to alleviate the insufficient training. Specifically, when the architecture weights $\overline{\alpha}$ are updated, the architecture of the evaluation network will be changed accordingly. The weights of the new evaluation network are initialized with the parameters inheriting from previous training. In other words, the evaluation network has a one-shot model that shares weights between architectures that have common edges. This speeds up the mature convergence of the new evaluation network, thus elevating its capacity on feature representation. This weight-sharing strategy is different from that in single-path one-shot approaches [20], [77], which performs random sampling for architecture selection. In contrast, we select architectures to be optimized by the search network, which alleviates the issue of unbalanced training in prior methods [77], [91].

3.3 Network Architecture

The network architecture of the proposed CDARTS is presented in Fig. 2. It consists of two branches: a search network with a few stacked cells and an evaluation network with dozens of cells. Note that the search and evaluation networks share the same architectures with previous DARTS approaches [7], [15], [38], [81].

For information transfer, we build up connections between the two branches. Concretely, there is an *architecture transfer* path delivering the discovered cell motifs from the search branch to the evaluation branch, as the top bold

TABLE 1
The Hyperparameters Used in Search

Benchmark	BS	LR	OPT	MOME	WD	S_S	S_E	S_U	λ	λ'	Runs
NATS-Bench [95]	64	0.1	SGD	0.9	3e-4	$50s$	$1s$	$1s$	4	5	3
CIFAR10	64	0.1	SGD	0.9	3e-4	$30s$	$1s$	$1s$	4	5	5
ImageNet	1024	0.8	SGD	0.9	3e-5	$30s$	$3s$	$1s$	4	5	5

BS, LR, OPT, MOME and WD are the batch size, learning rate, optimizer, momentum and weight decay. λ and λ' are the coefficients in Eqs. (8) and (11). Runs means the number of independent runs. The unit s of S_S, S_E and S_U in Algorithm 1 represents the number of steps in one epoch.

arrow presented in Fig. 2. Note that the cell structure discovered by the search network is a fully connected graph. In other words, all the candidate operations are applied to calculate the feature of each node in the graph, i.e., Eq (1). When using this continuous cell structure to construct a new evaluation network, we need to conduct discretization first. Following previous works [7], we retain the top-k ($k = 2$) strongest operations among all the candidates from all the previous nodes. This derived discrete cell structure serves as the basic building block for the evaluation branch.

On the other hand, there is another *introspective distillation* path transferring the feature feedback of the evaluation branch to the search branch, as the bottom solid arrow shown in Fig. 2. The feedback serves as the supervision signal for the search network to find better cell structures. In details, we use multi-level features of the evaluation network as the feedback signals, since they are representative on capturing image semantics. As the lateral *embedding* connections presented in Fig. 2, the multi-level features combine low-resolution, semantically strong features with high-resolution, semantically weak features. The features are derived from the outputs of each stage, and then passed through an *embedding* module to generate the corresponding feature logits. The function of the *embedding* module is to project the dense feature maps into a low dimensional subspace. The obtained logits of the evaluation network is used as the supervision signals for the search network through a soft cross-entropy layer, as in Eq. (9).

4 Experiments

We evaluate the proposed CDARTS algorithm on the image classification task and conduct a series of experiments on CIFAR [29], ImageNet [92], as well as the recent proposed NATS-Bench [95] benchmark [31]. Furthermore, we apply our introspective distillation search method into the chain-structured search space [12] to verify the generalization capability of our method.

4.1 Implementation Details

In Table 1, we elaborate the setting of the hyperparameters used in CDARTS on different benchmarks. The other settings are the same as DARTS families [7], [15], [81]. In line with prior works [7], [15], [81], the architectures found on CIFAR10 and ImageNet need to be trained from scratch.

CIFAR10. To train the networks, we divide the 50K training images of CIFAR10 [29] into two equal parts. One part serves as the *train* set to learn the weight w_S of the search network, while the other part works as the *val* set to optimize the architecture hyperparameter α and the evaluation network weights w_E. In the search phase, w_E^* is only optimized on the validation set. After the search, we retrain the discovered architectures on both train and validation sets for a fair comparison with other DARTS methods. During training, the total number of search step S_S is set to 30 epochs, the evaluation step S_E and the update step S_U are both set to 1 epoch. When training the weight w_S and w_E individually, the batch size, learning rate, momentum and weight decay are set to 64, 0.1, 0.9 and 3×10^{-4}, respectively. When jointly updating α and w_E, we adopt the Adam optimizer [93] with a fixed learning rate of 3×10^{-4}. The momentum and weight decay are set to $\{0.5, 0.99\}$ and 3×10^{-4}, respectively. The coefficient λ is fixed to 4, while λ' decays from 5 to 0 in the first 20 epochs and remains at 0 in the last 10 epochs.

ImageNet. ImageNet [92] is much larger than CIFAR [29] in both scale and complexity. In line with prior works, i.e., PCDARTS [81] and DARTS+ [22], we randomly sample 10% images from ImageNet to form the *train* data, while sampling another 10% images to form the *val* data. The construction of networks tested on ImageNet is similar to the one on CIFAR, but has two differences. First, to align with previous works [7], [15], [81] on ImageNet, we set the number of cells to 8 and 14 for the search and evaluation networks. Second, following the same setting as DARTS [7], [15], [81], the stem layer contains 3 convolution operations reducing the feature size from 224×244 to 28×28. During search, we first pre-train the search network with 10 epochs when the architecture hyperparameters are fixed. Then, the number of search step S_S is set to 30 epochs, while the evaluation step S_E is set to 3 epochs. When training the weight w_S and w_E individually, the batch size, learning rate, momentum and weight decay are set to 1024, 0.8, 0.9 and 3×10^{-5}, respectively. When jointly updating α and w_E, we adopt the Adam optimizer [93] with a fixed learning rate of 3×10^{-4}. The momentum and weight decay are set to $\{0.5, 0.99\}$ and 3×10^{-5}, respectively. The coefficient λ is fixed to 4, and λ' decays from 5 to 0 in the first 20 epochs and remains at 0 in the rest epochs. We use 8 NVIDIA Tesla V100 GPUs with a batch size of 1024 to search on ImageNet. It takes about 5 hours with our PyTorch [94] implementation.

4.2 Evaluation on NAS Benchmark

NATS-Bench benchmark [95] includes the search space of 15,625 neural cell candidates for architecture topology and 32,768 for architecture size on three datasets. We conduct our experiments on the topology search space S_t, covering all possible architectures generated by the fixed search space of 4 nodes and 5 associated operation options. It evaluates all the architectures on CIFAR10 [29], CIFAR100 [29] and ImageNet-16-120 [98], and provides the corresponding performance. It also benchmarks 13 recent NAS algorithms on the search space using the same setup for a fair comparison.

TABLE 2
Comparison With 10 NAS Methods Provided by NATS-Bench Benchmark [95] Topology Search Space S_t

Method	CIFAR10		CIFAR100		ImageNet-16-120	
	validation	test	validation	test	validation	test
ResNet [62]	90.83	93.91	70.50	70.89	44.10	44.23
Optimal	91.61	94.37(94.37)	73.49	73.51(73.51)	46.73	46.20(47.31)
REA [66]	91.25±0.31	94.02±0.31	72.28±0.95	72.23±0.84	45.71±0.77	45.77±0.80
REINFORCE [34]	91.12±0.25	93.90±0.26	71.80±0.94	71.86±0.89	45.37±0.74	45.64±0.78
RANDOM [96]	91.07±0.26	93.86±0.23	71.46±0.97	71.55±0.97	45.03±0.91	45.28±0.97
BOHB [97]	91.17±0.27	93.94±0.28	72.04±0.93	72.00±0.86	45.55±0.79	45.70±0.86
ENAS [35]	90.20±0.00	93.76±0.00	70.21±0.71	70.67±0.62	40.78±0.00	41.44±0.00
SPS [20]	87.60±0.61	91.05±0.66	68.27±0.72	68.26±0.96	39.37±0.34	40.69±0.36
SETN [33]	90.02±0.97	92.72±0.73	69.19±1.42	69.36±1.72	39.77±0.33	39.51±0.33
DARTSV1 [7]	49.27±13.44	59.84±7.84	61.08±4.37	61.26±4.43	38.07±2.90	37.88±2.91
DARTSV2 [7]	58.78±13.44	65.38±7.84	59.48±5.13	60.49±4.95	37.56±7.10	36.79±7.59
GDAS [18]	89.69±0.72	93.23±0.58	68.35±2.71	68.17±2.50	39.55±0.00	39.40±0.00
CDARTS	91.12±0.44	94.02±0.31	72.12±1.23	71.92±1.30	45.09±0.61	45.51±0.72

Optimal *indicates the best performing architecture in the search space.*

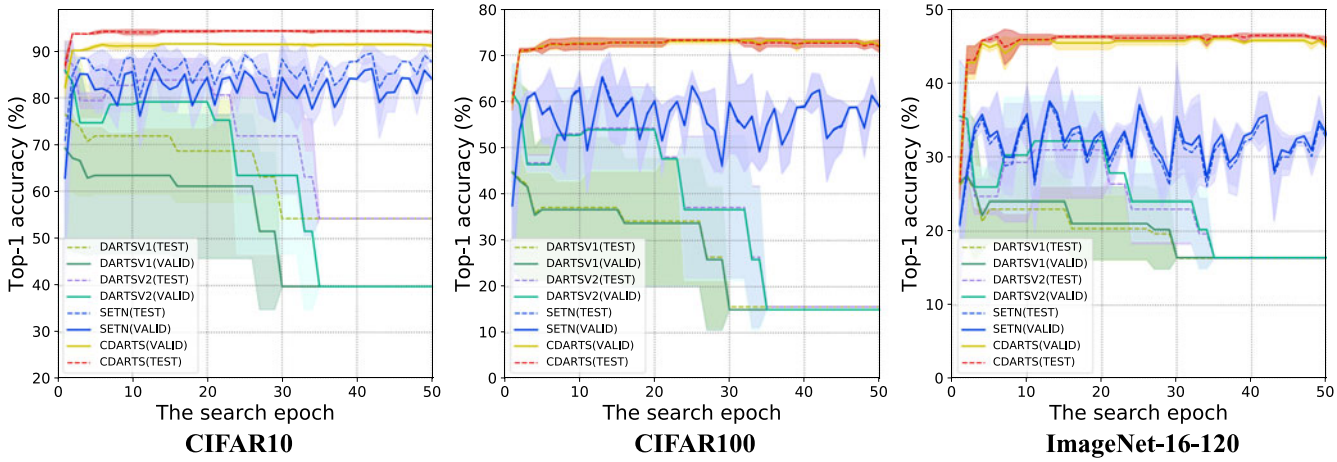
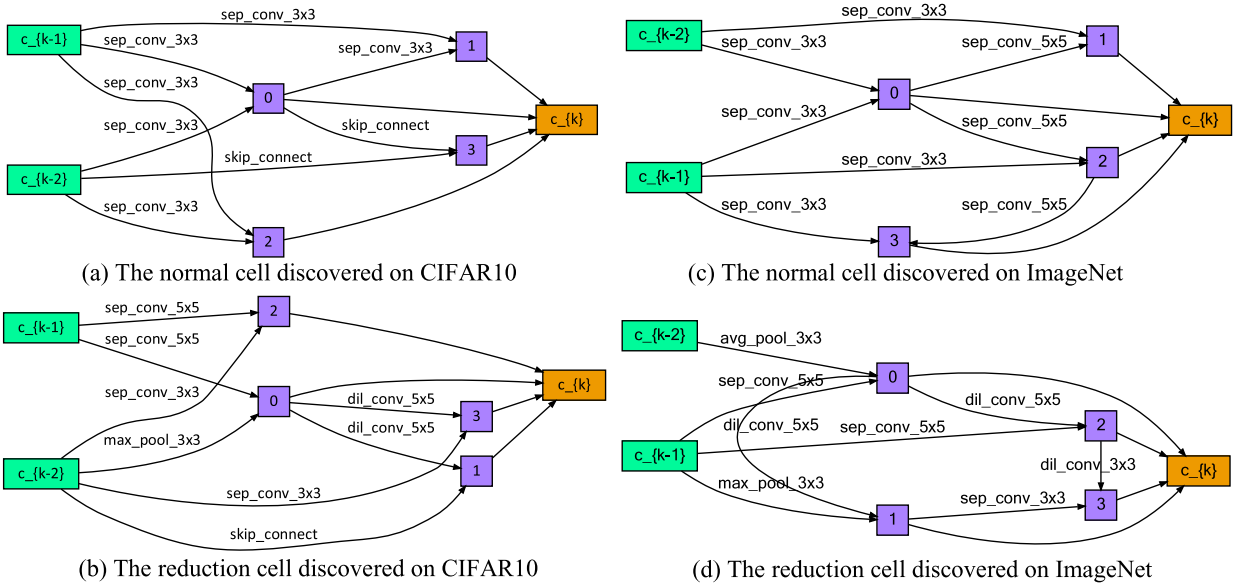


Fig. 3. Test and validation accuracy ($mean \pm std$) versus search epochs on NATS-Bench benchmark [95] topology search space S_t. DARTSV1 and DARTSV2 indicate the first-order and second-order DARTS methods [7], whose results are provided by the benchmark [95].

According to the evaluation rules of NATS-Bench benchmark [95], i.e., *no regularization for a specific operation* and *report results of multiple searching runs*, we remove the ℓ_1-norm regularization imposed on skip-connections, i.e., Eq. (11), and report the results of three independent runs. All the DARTS-based methods perform 50 searching epochs, following the same setting as [95].

We first compare our method with the 10 NAS methods that have been benchmarked on NATS-Bench benchmark [95], such as the evolution algorithm based REA [66] and the one-shot based ENAS [35]. As presented in Table 2, our CDARTS outperforms 9/10 methods on the three datasets. It achieves comparable performance to the REA method [66], while searching much faster. Both of REA and CDARTS perform superior to the human-design ResNet [62], being close to theoretical optimum on the benchmark, i.e., the "Optimal" in Table 2. The original DARTS underperforms due to overfitting [95], and its standard deviation of performance is 0 because the final found architecture has plenty of skip-connections. The underlying reason is that DARTS tends to overfit to the architectures with many skip-connection operations. GDAS [18] is another DARTS based method. It performs much better than DARTS on NATS-Bench benchmark because it introduces an architecture sampling optimization algorithm to prevent overfitting. In contrast, our CDARTS can perform well-balanced and achieves superior performance. The underlying reason is due to the proposed unified architecture, which allows the search network to discover architectures tailored for the target evaluation network. Except for CDARTS, the performance of all the NAS approaches in Table 2 are provided by the official NATS-Bench benchmark [95].

We conduct another experiment to evaluate the searching stability. With the NATS-Bench benchmark [95] benchmark, it is easy to track the validation and test accuracy of every discovered architecture after each training epoch, as visualized in Fig. 3. We can see that DARTS [7] achieves a relatively high accuracy at early stage, however, as the search process continues, its accuracy drops significantly and the stability becomes weak. Finally, DARTS falls into the overfitting status, in which the final architecture contains many skip-connections. In contrast, the performance of our CDARTS is improved gradually along with the increase of search epochs, and eventually reaches stable and converged. This



(a) The normal cell discovered on CIFAR10

(c) The normal cell discovered on ImageNet

(b) The reduction cell discovered on CIFAR10

(d) The reduction cell discovered on ImageNet

Fig. 4. Cells discovered on CIFAR10 and ImageNet. ImageNet cells are deeper than CIFAR10.

may attribute to the knowledge transfer between the search and evaluation networks. CDARTS continually leverages the supervision from the evaluation network to guide the search process, thus preventing the searched architecture from collapsing. Moreover, it is observed that the one-shot NAS method SETN [33] performs inferiorly to our method in term of both accuracy and searching stability.

4.3 Evaluation on CIFAR

The CIFAR image classification dataset contains two subsets, CIFAR10 and CIFAR100. CIFAR10 has 10 classes. Each class consists of 6,000 images, in which 5,000 images are used for training and 1,000 for testing. CIFAR100 consists of 100 classes. Each class containing 600 images, where 500 images are used for training and the rest for testing.

We search the architecture on CIFAR10 test set, and evaluate it on both CIFAR10 and CIFAR100. The cell topologies discovered on CIFAR10 are shown in Fig. 4. After discovering the evaluation network architecture, we retrain it by following the same setting as PDARTS [15]. Specifically, the number of channels is set to 36 and the structure is the same with the search stage. We use the entire CIFAR training images to train the network from scratch with 600 epochs. To speed up the training, the batch size is set to 128 and the learning rate decays from 0.025 to 0 with a cosine annealing [103]. We choose SGD [104] as the optimizer with a momentum of 0.9 and weight decay of 5×10^{-4}. In line with PDARTS, the drop-path [105] rate is set to 0.3, the auxiliary towers [106] is set to 0.4, and the cutout regularization [107] length is set to 16. The experiments are executed on one NVIDIA Tesla V100 GPU.

We report the performance of five individual runs of different search algorithms in Table 3. In the five independent runs, CDARTS consistently outperforms prior DARTS [7] and PCDARTS [81]. Moreover, in terms of performance deviations, CDARTS performs better than prior methods, i.e., CDARTS: $97.52 \pm 0.04\%$ versus DARTSV1 [7]: $96.93 \pm 0.13\%$, DARTSV2: $96.85 \pm 0.29\%$, PC-DARTS: $97.33 \pm 0.07\%$. The cell motifs discovered on CIFAR-10 are presented in Fig. 4. We observe that the network prefers to choose separable convolutions [113], and is therefore capable of improving the model capacity and serves as a key component for network construction. Besides, these cells usually contain either one or two skip-connections, and the depth of these cells are usually two (the longest path from input to output node).

The performance of discovered cells are reported in Table 4. It is observed that our CDARTS achieves superior performance to some ConvNets designed manually or automatically. For instance, CDARTS surpasses the manually designed DenseNet-BC [63] by 1.18 and 3.93 points on CIFAR10 and CIFAR100, respectively. Compared with the original DARTS method [7], CDARTS also achieves better performance ($97.52 \pm 0.04\%$ versus $97.00 \pm 0.14\%$). CDARTS is also slightly superior to the recently proposed PDARTS [15] and PCDARTS [81] on both CIFAR10 and CIFAR100. It is worthy noting that the performance of ProxylessNAS [14] is slightly better than CDARTS, but at the cost of a larger model parameter size and longer model search time. Moreover, compared with other non-gradient based methods, such as reinforcement learning (RL) or evolutionary algorithms, our method is also competitive. For example, the RL-based ENAS [35] method achieves 97.11% test accuracy on CIFAR10, which is slightly inferior to 97.52% of CDARTS.

4.4 Evaluation on ImageNet

ImageNet [92] is a large-scale image classification dataset. It consists of 1,000 categories with 1.2 million training images

TABLE 3
Top-1 Accuracy of Searched Architectures in Five Independent Search Runs on CIFAR10

Methods	Runs					Mean ± std
	#1	#2	#3	#4	#5	
DARTSV1(%)	97.11	96.85	97.01	96.93	96.73	96.93±0.13
DARTSV2(%)	96.89	96.32	97.23	96.86	96.94	96.85±0.29
PCDARTS(%)	97.28	97.33	97.43	97.25	97.36	97.33±0.07
CDARTS(%)	**97.53**	**97.45**	**97.60**	**97.52**	**97.54**	**97.52±0.04**

TABLE 4
Comparison With SOTA Architectures on CIFAR10 and CIFAR100

Architecture	Test Acc. (%)		Param	Latency	Search Cost	Search
	CIFAR10	CIFAR100	(M)	(ms)	(GPU days)	Method
Wide ResNet [99]	96.20	82.70	36.5	-	-	manual
ResNeXt-29, 16x64d [100]	96.42	82.69	68.1	-	-	manual
DenseNet-BC [63]	96.52	82.82	25.6	-	-	manual
NASNet-A [1]	97.35	-	3.3	-	1800	RL
AmoebaNet-B [66]	97.45	-	2.8	-	3150	evolution
PNAS [10]	96.59	-	3.2	-	225	SMBO
ENAS [35]	97.11	-	4.6	-	0.5	RL
NAONet [101]	96.82[1]	84.33	10.6	-	200	NAO
SNAS (moderate) [17]	97.15 ± 0.02	-	2.8	30.2	1.5	gradient
ProxylessNAS [14]	97.92	-	5.7	-	4	gradient
GDAS [18]	97.07	81.62	3.4	30.6	4	gradient
Random†	96.75 ± 0.18	-	3.4	-	-	-
DARTSV1 [7]†	97.00 ± 0.14	82.24	-	3.3	1.5	gradient
DARTSV2 [7]†	97.24 ± 0.09	82.46	40.9	3.3	4.0	gradient
PDARTS [15]†	97.50	83.45	3.4	40.9	0.3	gradient
PCDARTS [81]†	97.43 ± 0.06	-	3.6	40.7	0.1	gradient
FairDARTS [38]†	97.41 ± 0.14	-	3.8	-	0.1	gradient
LA-DARTS [102]†	97.28 ± 0.05	-	2.7	28.4	0.7	gradient
CDARTS†	97.52 ± 0.04	84.31	3.9 ± 0.08	41.2 ± 0.5	0.3	gradient

† *indicates using the DARTS [7] search space. Latency is measured on an NVIDIA Tesla-P100 GPU with a batch size of 32 and an input size of 32×32.*

and 50K validation images. Note that, for a fair comparison, the images in ImageNet are resized to 224×224 pixels in our experiments in line with prior DARTS works [7], [15], [38], [81]. Once the search process is completed, we retrain the discovered architecture following the same setting as PDARTS. The final evaluation network contains 14 cells with a channel number of 48. We retrain it for 250 epochs from scratch with full ImageNet training data. A standard SGD [104] optimizer is adopted with a momentum of 0.9 and weight decay of 3×10^{-5}. The learning rate is set to 0.5 with the cosine scheduler; meanwhile, a learning rate warmup [114] is applied in the first 5 epochs. Same as DARTS, the label smoothing [115] ratio is set to 0.1 and the auxiliary tower is set to 0.4.

The evaluation results of the searched architectures are reported in Table 5. It shows that the architectures discoverd by our CDARTS is slightly better than the manually designed models, such as MobileNet-V2 [60] and ShuffleNet [116]. Compared with automated methods, *e.g.*, RL-based MobileNet-V3 [32], our gradient-based CDARTS achieves comparable performance. MobileNet-V3 blends automatic search techniques with the interaction of human design, while CDARTS is purely automatic. Moreover, CDARTS outperforms the prior DARTS [7] by 3.0 points in term of top-1 accuracy. This improvement is attributed to the proposed search algorithm. It is observed that the *flops* of CDARTS is a little higher than other DARTS methods. This is solely caused by the search algorithm itself, because the search space is the same among these DARTS methods. To follow the mobile setting [7] comparison, i.e., the number of multiply-add operations in the model is restricted to be less than $600M$, we reduce the number of model channels from 48 to 44, while keeping other settings unchanged. We obtain 75.9% top-1 accuracy with $571M$ *flops*, which is still slightly superior than PDARTS [15] and PCDARTS [81]. In addition, when training the discovered CIFAR10 cells on ImageNet, CDARTS obtains a top-1 accuracy of 75.6%, which is on par with ImageNet cell 76.3%. This demonstrates the generalization potential of CDARTS.

In Table 6, we present the results of five independent searches on ImageNet. We observe that the five runs of our method consistently exceed that of PDARTS (75.6%) and PCDARTS (75.8%). The discovered cells on ImageNet are presented in Fig. 4. We observe that the cells discovered on ImageNet is deeper than the ones on CIFAR10, because the classification on ImageNet is more complex. This is aligned with the evidence that increasing network depth is beneficial for elevating model capability [62]. Moreover, the discovered cells on ImageNet contain larger convolution kernels (i.e., 5x5 sep_conv), which is helpful to improve model capacity.

4.5 Ablation Study

Component-wise Analysis. To further understand the effects of the components in the proposed search algorithm, we test four variations of CDARTS on the ImageNet dataset. In particular, each variation corresponds to an optimization objective listed in Table 7, and parameters for each model are tuned separately to obtain the optimal results. It is worth noting that the alternating optimization of $\mathcal{L}^{S}_{train} + \mathcal{L}^{S}_{val}$ fail to search on ImageNet, whose cells are full of skip-connections. So we use the ℓ_1-norm regularization to stabilize the searching process, and achieve a top-1 accuracy of 72.8%. By adding the proposed joint learning $\mathcal{L}^{S,E}_{val}$ into optimization, the performance is improved to 75.7%. This indicates the effectiveness of the multi-level feature semantics guidance of the evaluation network, and the integration of the search and evaluation is beneficial to discover more robust architecture. Moreover, during joint training, updating the weights of the evaluation network can further improve the performance by 0.9%.

TABLE 5
Results on ImageNet

Architecture	Test Acc. (%)		Params	×+	Search Cost	Search Method
	Top-1	Top-5	(M)	(M)	(GPU days)	
Inception-V1 [108]	69.8	89.9	6.6	1448	-	manual
SqueezeNext [109]	67.5	88.2	3.23	708	-	manual
MobileNet-V2 (1.4×) [60]	74.7	-	6.9	585	-	manual
ShuffleNet-V2 (2×) [110]	74.9	-	7.4	591	-	manual
NASNet-A [1]	74.0	91.6	5.3	564	1800	RL
AmoebaNet-C [66]	75.7	92.4	6.4	570	3150	evolution
PNAS [10]	74.2	91.9	5.1	588	225	SMBO
EfficientNet-B0 [36]	77.1	93.2	5.3	390	-	RL
MnasNet-92 [57]	74.8	92.0	4.4	388	-	RL
SPOS [77]	74.3	-	-	319	-	evolution
FairNAS-A [79]	75.3	92.4	4.6	388	-	evolution
MobileNet-V3 [32]	76.6	-	7.5	356	-	RL
MoGA-A [78]	75.9	92.8	5.1	304	12	evolution
SNAS (mild) [17]	72.7	90.8	4.3	522	1.5	gradient
XNAS [111]	76.0	–	5.2	–	0.3	gradient
ProxylessNAS [14]	75.1	92.5	7.1	465	8.3	gradient
ASAP [112]	73.3	-	-	-	0.2	gradient
DARTS [7]†	73.3	91.3	4.7	574	4.0	gradient
FairDARTS [38]†	75.6	92.6	4.3	440	3.0	gradient
PDARTS [15]*†	75.6	92.6	4.9	557	0.3	gradient
PCDARTS [81]*†	75.8	92.7	5.3	597	3.8	gradient
EnTranNAS (ImageNet) [16]*†	75.7	92.8	5.5	637	1.9	gradient
CDARTS(MS)*†	75.9	92.6	5.4	571	1.7	gradient
CDARTS*†	76.3 ± 0.3	92.9 ± 0.2	6.1 ± 0.2	701 ± 32	1.7	gradient

† *We use the same search space as DARTS [7].* ***MS*** *denotes mobile setting.* * *denotes use 10% of ImageNet data for searching.*

Correlation Analysis. In differentiable architecture search methods, the operations and edges with weak attention (i.e., small weights) are considered as redundant and are pruned to obtain a compact architecture (i.e., top-k discretization). However, it is not clear whether the operations and edges with weak attention are truly redundant, while strong attention indicates the high importance. Therefore, we conduct another experiment to evaluate the correlation between the architecture hyperparameter and the true performance of architectures. Specifically, we sample a variety of cell architectures and rank them according to the learned weights of the architecture hyperparameter. The quantitative results shown in Fig. 5 demonstrate that our method obtains better correlation than DARTS [7]. We further compare the Kendall Rank Correlation Coefficient (τ) metric that evaluates the rank correlation of data pairs. We repeat the experiment six times with different seeds. The Kendall's τ of SPOS [77] is 0.19 [89], while ours is 0.49, being 0.3 point superior to SPOS. To some extent, the architecture in CDARTS is able to reflect the relative ranking of architectures. But it is worth noting

TABLE 6
Evaluation of Searched Architectures in Five Independent Search Runs on ImageNet

Accuracy	Runs					Mean ± std
	#1	#2	#3	#4	#5	
Top1(%)	76.45	75.90	**76.61**	76.40	75.94	76.26±0.29
Top5(%)	92.83	92.80	**93.20**	93.03	92.75	92.92±0.17
Params(M)	6.31	5.99	6.36	6.32	5.73	6.14±0.24
Flops(M)	718.68	695.98	725.20	717.22	651.45	701.70±26.97

TABLE 7
Ablation Study on ImageNet

$\mathcal{L}^S_{train} + \mathcal{L}^S_{val}$	✓	✓	✓	✓
$+ \mathcal{L}_{Reg}$		✓	✓	✓
$+ \mathcal{L}^{S,E}_{val}$			✓	✓
$+ \mathcal{L}^{E}_{val}$				✓
Top-1 Acc. (%)	-	72.8	75.7	76.6
Params. (M)	-	4.54	5.83	6.36
Flops (M)	-	481.58	675.52	725.20

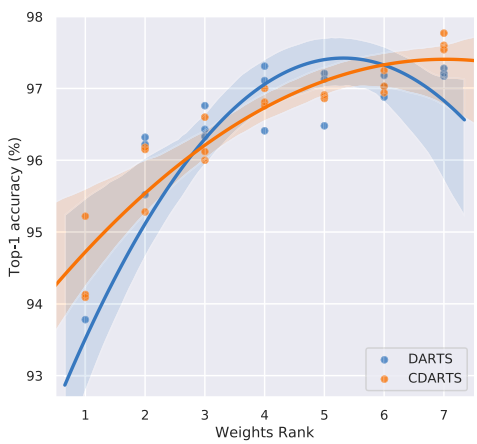


Fig. 5. Operation weights and retrain accuracy correlation analysis. Weights Rank indicates the weight sorting of the learned architecture from small to large.

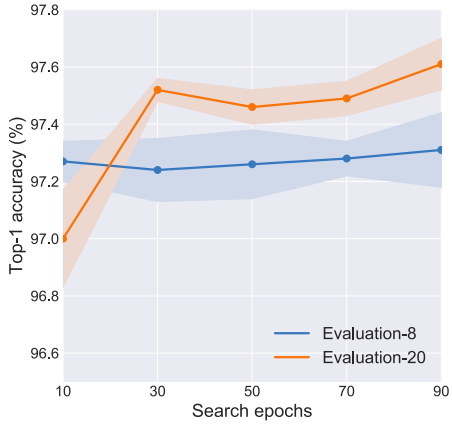


Fig. 6. Ablation of the search epochs and the depth of evaluation network. Evaluation-8/20 represent the evaluation networks with 8/20 cells, respectively.

that CDARTS still cannot well distinguish the architectures that share close performance.

Depth of Evaluation Network. Due to the limitation of GPU memory, the search network of DARTS can only stack 8 cells, while the evaluation network contains 20 cells. This brings the so-called *depth gap* issue studied in PDARTS [15]. Such problem is not observed in our method, and we believe the reason is due to the integration of search and evaluation in a unified architecture. As shown in Fig. 6, we compare the performances of different numbers of cells in the evaluation network. It clearly shows that the 20-cell evaluation network (the red line) performs better than the 8-cell network (the green line). This indicates the proposed jointly training of the two networks mitigates the impacts of the depth gap.

Impact of Search Epoch. In DARTS methods [7], [81], after certain searching epochs, the number of skip-connections increases dramatically in the selected architecture, which results in the search collapse [22]. We study the impact of the number of search epochs in our CDARTS. From Fig. 6, we can see that when the number of search epochs approaches 30, the performance becomes saturated, and the structure of the evaluation network tends to be stable. So the search epoch is set to 30 in experiments. In addition, as the number of search epochs increases, CDARTS continues to search for better structures. Our method even achieves 97.77% top-1 accuracy on CIFAR10 when the search epochs are 90.

4.6 Generality and Robustness

4.6.1 Extension to Chain-Structured Search Space

A robust search algorithm should be capable of searching architectures over diverse search spaces. We migrate the introspective distillation searching method to a chain-structured search space [12]. The search space consisting of mobile inverted bottleneck MBConv [60] and squeeze-excitation modules [64], which is the same as recent works [28], [61] for fair comparisons. There are seven basic operators, including MBConv with kernel sizes of {3,5,7} and expansion rates of {4,6}, and an additional skip connection to enable elastic depth of architectures. The space contains about 3.69×10^{16} architecture candidates in total. For simplicity, we set SPOS [77] as our baseline method and introduce a simple principle to select the teacher path. Specifically, the teacher path is first initialized with random paths within the search network. Then, for each batch, we randomly sample a single path from the search network and train the path under both the teacher path soft-label supervision and the ground truth supervision. After that, we evaluate the path on the validation dataset (a subset is used to save computation), and get its performance. If the current path performs superior to the teacher path and with less or the same *Flops*, then the teacher network will be replaced by the current path. In this way, the final left teacher path is the optimal path of all sampled paths during the search.

We train the search network for 120 epochs using the following settings: the SGD optimizer [104] with momentum 0.9 and weight decay 4e-5, initial learning rate 0.5 with linear annealing. The discovered architectures are presented in Fig. 7 and we retrain them for 350 epochs on ImageNet using similar settings as EfficientNet [36]: RMSProp optimizer with momentum 0.9 and decay 0.9, weight decay 1e-5, dropout ratio 0.2, initial learning rate 0.04 with a warmup [114] in the first 3 epochs and a cosine annealing. We also adopt the AutoAugment [120] and exponential moving average techniques. We use 8 Nvidia Tesla V100 GPUs with a batch size of 1,024 for the retraining.

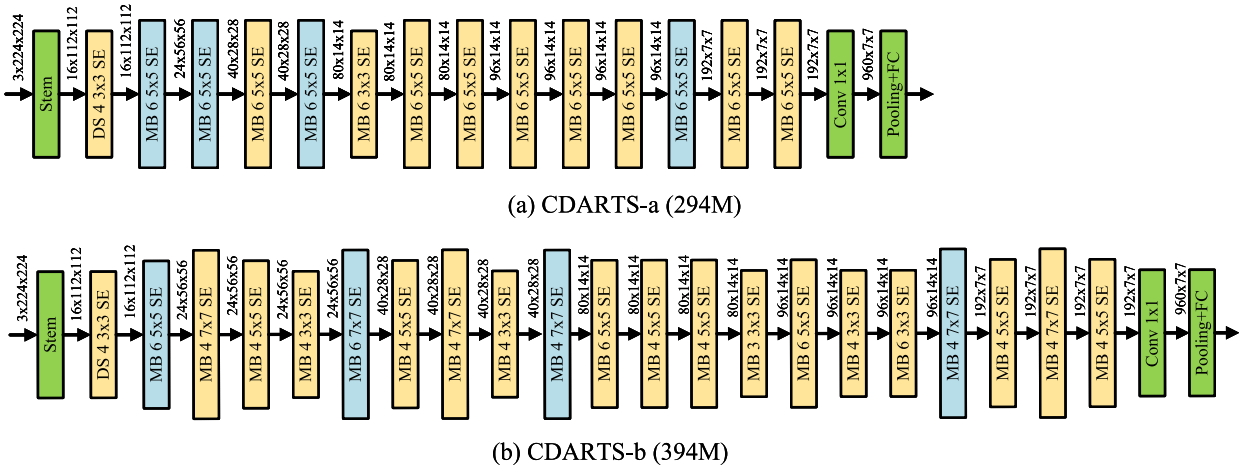


Fig. 7. Discovered architectures. "$MB\ a\ b \times b$" represents the inverted bottleneck MBConv with the expand rate of a and kernel size of b. "$DS\ a\ b \times b$" denotes the depthwise separable convolution with the expand rate of a and kernel size of b.

TABLE 8
Comparisons of Chain-Structured Search Space Models on ImageNet

Model	Params	FLOPs	Acc@1	Search
SPOS† [77]	3.5M	319M	74.3%	12
DSNAS† [117]	3.5M	324M	74.4%	17.5
FBNet-C [76]	5.5M	375M	74.9%	9
MobileNetV3♯ [32]	5.3M	219M	75.2%	1667
MnasNet-A3♯ [57]	5.2M	403M	76.7%	1667
OFA♯† [61]	5.8M	230M	76.9%	55
FairNAS-A† [79]	4.6M	388M	75.3%	12
MoGA-A♯† [78]	5.1M	304M	75.9%	12
PC-NAS-S† [80]	5.1M	-	76.8%	-
MixNet-M♯† [118]	5.0M	360M	77.0%	1667
DNA-b♯† [28]	4.9M	406M	77.5%	24.6
EfficientNet-B0♯ [36]	5.3M	399M	77.1%	-
ProxylessNAS† [14]	7.1M	465M	75.1%	15
SCARLET-A† [119]	6.7M	365M	76.9%	10
CDARTS-a♯†	7.0M	294M	77.4%	12
CDARTS-b♯†	6.4M	394M	**78.2%**	12

The input size is 224 × 224. ♯: using the SE module. The unit of search time is GPU day. †: the search space contains kernel-size-7 operation.

It is worth noting that there are a few recent works that leverage knowledge distillation techniques to boost searching [28], [61]. As shown in Table 8, our introspective distillation shows superior performance over these methods. Specifically, DNA-b [28] recruits EfficientNet-B7 [36], a very high-performance third-party model, as the teacher and achieves 77.5% top-1 accuracy, while our method (CDARTS-b) gets a superior accuracy of 78.2% without using any pre-trained model. Our model also surpasses EfficientNet-B0 by 1.1% with nearly the same model size. The leading performance indicates that the proposed introspective distillation method also works well on the chain-structured search space.

4.6.2 Extension to Big Model

To further unleash the power of the searched architectures, we enlarge the evaluation network channels and train from scratch with more data augmentations [120], [121], which is denoted as *big* model.

Big Model on CIFAR10 We enlarge the numbers of feature channels from 36 to 50. After 2000 epochs training, the CDARTS-BIG model obtains impressive performance improvements. On CIFAR10, the test accuracy increases from 97.52% to 98.32%. This improvement further verifies the effectiveness of the proposed method. In addition, to have a fair comparison, we retrain the evaluation networks discovered by other DARTS methods, such as DARTS, PDARTS and PCDARTS with the same *big* setting. As presented in Table 10, our CDARTS performs the best among them.

Big Model on ImageNet We increase the channel number from 48 to 96 and the input image size from 224×224 to 320×320 to construct a big model with $5.6B$ *flops*. This big model is trained for 250 epochs, and obtains a 4.86 points absolute improvement over the original CDARTS, achieving 81.12 top-1 accuracy on ImageNet, which is comparable to EfficientNet-B2 [36]. Moreover, following the same settings, we train the big models of PDARTS and PCDARTS. Their performances are inferior to our CDARTS, as presented in Table 10. It should be noted that the original DARTS cells have been proved in many experiments to be inferior to the improved version, e.g., PDARTS [15], PCDARTS [81]. In order to save computation resources, we did not train big model of DARTS on ImageNet. These results confirm again the generalization potential and effectiveness of CDARTS.

4.6.3 Extension to Object Detection

We also transfer the proposed method to the downstream object detection task, which is a fundamental task in the vision community. We use the discovered architecture and corresponding weights pre-trained on ImageNet as a drop-in replacement for the backbone feature extractor in RetinaNet [132]. The capabilities of various light-weight backbones are compared on the COCO benchmark [133]. We fine-tune the searched model on the *train2017* set (118k images) and evaluate on the *val2017* set (5k images) with the batch size of 16 on 8 V100 GPUs. Similar to [79], the schedule is the default 1×, i.e., 12 epochs. The initial learning rate is set to 0.02, and then decayed with the scaling factor of 0.1 at the 8th and 11th epoch. The optimizer is SGD with momentum 0.9 and weight decay 1e-4. We use the MMDetection toolbox [134] based on PyTorch [94].

Results are summarized in Table 9. Specifically, CDARTS-a surpasses MobileNetV2 by 6.9% while using fewer Flops. Compared to MnasNet [57], CDARTS-a utilizes 19% fewer Flops yet achieves 4.7% higher performance. CDARTS-b even achieves 36.2% mAP on the COCO val2017, which surpasses other methods by a large margin. With similar flops and

TABLE 9
Object Detection Results of Various Drop-in Backbones on the COCO val2017

Backbones	Input Size	FLOPS(G)	Params(M)	mAP(%)	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	Top-1(%)
MobileNetV2† [60]	1280×800	6.1	3.4	28.3	46.7	29.3	14.8	30.7	38.1	72.0
SPOS† [77]	1280×800	7.4	4.3	30.7	49.8	32.2	15.4	33.9	41.6	75.0
MobileNetV3† [32]	1280×800	4.5	-	29.9	49.3	30.8	14.9	33.3	41.1	75.2
MnasNet-A2† [57]	1280×800	6.9	4.8	30.5	50.2	32.0	16.6	34.1	41.1	75.6
MixNet-M† [118]	1280×800	7.3	5.0	31.3	51.7	32.4	17.0	35.0	41.9	77.0
FairNAS-A† [79]	1280×800	8.0	5.9	32.4	52.4	33.9	17.2	36.3	43.2	77.5
MixPath-A [122]	1280×800	7.1	5.0	31.5	51.3	33.2	17.4	35.3	41.8	76.9
CDARTS-a	1280×800	6.0	7.0	35.2	55.5	37.5	19.8	38.7	47.5	77.4
CDARTS-b	1280×800	8.1	6.4	**36.2**	**56.7**	**38.3**	**20.9**	**39.8**	**48.5**	**78.2**

Acc represents the top-1 accuracy on ImageNet. † reported by [79]

TABLE 10
Top-1 Accuracy of big Models

Method	CIFAR10	ImageNet
DARTS [7]	97.95	–
PDARTS [15]	98.00	80.04
PCDARTS [81]	97.92	79.81
CDARTS	**98.32**	**81.12**

parameters, CDARTS-b is 3.8% higher than FairNAS-A [79], suggesting the architecture has good generalization ability.

4.6.4 Extension to Semantic Segmentation

To evaluate the generalization ability of the architectures found by CDARTS, we transfer the architectures to the downstream semantic segmentation [46], [80], [135] task. We leverage the discovered architecture and corresponding weights pre-trained on ImageNet as a drop-in replacement for the encoder in FasterSeg [128]. The mean intersection over union per class (mIoU) is used as the metric for semantic segmentation.

Cityscapes [136] is a popular dataset which contains a diverse set of stereo video sequences recorded in street scenes from 50 different cities. It has 5,000 high quality pixel-level annotations. There are 2,975 images for training, 500 images for validation. And for testing, it offers 1,525 images without ground-truth for a fair comparison. The dense annotation contains 19 classes for each pixel. We evaluate our CDARTS on Cityscapes validation set with the original image resolution of 1024×2048. In Table 11, we see the superior mIoU and model size of our CDARTS. Without any inference trick, our CDARTS achieves 78.1% mIoU, which is 5.0% better than FasterSeg [128]. Specifically, our CDARTS surpasses SegFormer [37] by 1.9% with much fewer Flops.

ADE20K [137] has 20k images for training, 2k images for validation and 3k images for testing. It is used in ImageNet scene parsing challenge 2016, and has 150 classes and diverse scenes with 1,038 image-level. We also evaluate our CDARTS on the ADE20K validation set with the original image resolution of 640×640. From Table 12, we can see that our model is much smaller than others. Specifically, our CDARTS achieves 40.4% mIoU with 5.9G Flops, which is 3.0% better than SegFormer [37].

TABLE 11
Comparisons of Models for Semantic Segmentation on the Cityscapes Validation set

Methods	Encoder	Flops (G)	Params (M)	mIoU (%)
ESPNetV2 [123]	ESPNetV2	2.7	1.3	66.2
ICNet [124]	PSPNet50	-	-	69.5
HRNet [125]	HRNetV2	31.1	1.5	70.3
CAS [126]	CAS	-	-	71.6
MobilenetV3 [127]	Large	9.7	1.5	72.4
FasterSeg [128]	FasterSeg	28.2	4.4	73.1
BiSeNetV2-L [129]	-	21.2	-	73.4
SqueezeNAS [130]	LAT Large	19.6	1.9	73.6
SwiftNetRN-18 [131]	ResNet18	104.0	11.8	75.4
SegFormer [37]	MiT-B0	125.5	3.8	76.2
CDARTS(Ours)	CDARTS-b	20.7	5.9	**78.1**

TABLE 12
Comparisons of Models for Semantic Segmentation on the ADE20K Validation set

Methods	Encoder	Flops (G)	Params (M)	mIoU (%)
FCN [138]	MobileNetV2	39.6	9.8	19.7
PSPNet [139]	MobileNetV2	52.9	13.7	29.6
DeepLabV3+ [140]	MobileNetV2	69.4	15.4	34.0
SegFormer [37]	MiT-B0	8.4	3.8	37.4
CDARTS(Ours)	CDARTS-b	5.9	2.7	**40.4**

4.7 Discussion

Computation cost. The computation cost of CDARTS is comparable to DARTSV1. Both of them adopt the first-order optimization method [7] to train the networks. Compared with the original DARTS [7], we have an additional evaluation network and update it along with the search network in the search process. We denote the computation cost of updating the search network as $\mathcal{C}(|W_S|)$, which is the same as DARTSV1. In the search network cell, the number of edges between nodes is ED_S and the number of operations in each edge is OP_S. The number of stacked cells in the search network is N_S. Correspondingly, these factors of the evaluation network are denoted as ED_E, OP_E and N_E. Then the cost of the evaluation network $\mathcal{C}(|W_E|)$ is:

$$\mathcal{C}(|W_E|) = \frac{N_E \cdot ED_E \cdot OP_E}{N_S \cdot ED_S \cdot OP_S} \cdot \mathcal{C}(|W_S|) \tag{12}$$

In the experiments on CIFAR10, the ED_E, OP_E and N_E are set to 8, 1 and 20, respectively, while the ED_S, OP_S and N_S are set to 14, 8 and 8, respectively. Hence, the cost of the evaluation network $\mathcal{C}(|W_E|)$ is about $\frac{1}{6}$ of $\mathcal{C}(|W_S|)$. Compared to these two networks, the cost of the embedding modules is much smaller and can be ignored in cost estimation. Considering the initialization stage of the evaluation network (i.e., one training epoch for CIFAR10), the final cost of the evaluation network is about $\frac{1}{3}$ of $\mathcal{C}(|W_S|)$. In ImageNet, the final cost of the evaluation network is about $\frac{1}{2}$ of $\mathcal{C}(|W_S|)$. Therefore, compared to DARTSV1 [7], the complexity of CDARTS is only increased by $\sim$0.3 times.

Fast search speed on ImageNet. It is worth noting that our method takes about five hours to complete the search with eight GPUs on ImageNet. Such a fast search speed mainly attributes to the following three reasons. First, we use the fast first-order optimization algorithm [7] when updating the hyperparameter of architecture. Second, following PCDARTS [81], only 10% data in each category are used. Besides, the evaluation network is relatively lightweight and adopts the weight sharing strategy to speed up network training, so the extra computational cost of updating its parameters is small.

5 Conclusion

In this work, motivated by the separation problem of the search and evaluation networks in DARTS, we have proposed

a cyclic differentiable architecture search algorithm that integrates the two networks into a unified architecture. The alternating joint learning enables the search of architectures to fit the final evaluation network. Experiments on three different search spaces demonstrate the efficacy of the proposed algorithm and searched architectures, which achieve competitive performance on CIFAR, ImageNet and NATS-Bench [95]. In future work, we will consider adding more constraints on prioritized path selection, such as both Params and latency, thus improving the flexibility and user-friendliness of the search method.

REFERENCES

[1] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8697–8710.

[2] J. Chang *et al.*, "DATA: Differentiable architecture approximation with distribution guided sampling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 9, pp. 2905–2920, Sep. 2020.

[3] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "NAS-FPN: Learning scalable feature pyramid architecture for object detection," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 7036–7045.

[4] Y. Chen, T. Yang, X. Zhang, G. Meng, X. Xiao, and J. Sun, "DetNAS: Backbone search for object detection," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 6642–6652.

[5] C. Liu *et al.*, "Auto-deepLab: Hierarchical neural architecture search for semantic image segmentation," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 82–92.

[6] V. Nekrasov, H. Chen, C. Shen, and I. Reid, "Fast neural architecture search of compact semantic segmentation models via auxiliary cells," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9126–9135.

[7] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. Int. Conf. Learn. Representations*, 2019.

[8] P. Ren *et al.*, "A comprehensive survey of neural architecture search: Challenges and solutions," *ACM Comput. Surv.*, vol. 54, no. 4, pp. 1–34, 2021.

[9] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2017.

[10] C. Liu *et al.*, "Progressive neural architecture search," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 19–34.

[11] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. P. Xing, "Neural architecture search with Bayesian optimisation and optimal transport," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018.

[12] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 55, pp. 1–21, 2019.

[13] M. Wistuba, A. Rawat, and T. Pedapati, "A survey on neural architecture search," 2019, *arXiv:1905.01392*.

[14] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in *Proc. Int. Conf. Learn. Representations*, 2019.

[15] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proc. Int. Conf. Comput. Vis.*, 2019, pp. 1294–1303.

[16] Y. Yang, S. You, H. Li, F. Wang, C. Qian, and Z. Lin, "Towards improving the consistency, efficiency, and flexibility of differentiable neural architecture search," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 6667–6676.

[17] S. Xie, H. Zheng, C. Liu, and L. Lin, "SNAS: Stochastic neural architecture search," in *Proc. Int. Conf. Learn. Representations*, 2019.

[18] X. Dong and Y. Yang, "Searching for a robust neural architecture in four GPU hours," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1761–1770.

[19] X. Dong, M. Tan, A. W. Yu, D. Peng, B. Gabrys, and Q. V. Le, "AutoHAS: Efficient hyperparameter and architecture search," in *Proc. Int. Conf. Learn. Representations*, 2021.

[20] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," in *Proc. Uncertainty Artif. Intell.*, 2020, pp. 367–377.

[21] A. Zela, T. Elsken, T. Saikia, Y. Marrakchi, T. Brox, and F. Hutter, "Understanding and robustifying differentiable architecture search," in *Proc. Int. Conf. Learn. Representations*, 2020.

[22] H. Liang *et al.*, "DARTS+: Improved differentiable architecture search with early stopping," 2019, *arXiv:1909.06035*.

[23] R. Wang, M. Cheng, X. Chen, X. Tang, and C.-J. Hsieh, "Rethinking architecture selection in differentiable NAS," in *Proc. Int. Conf. Learn. Representations*, 2021.

[24] C. Sciuto, K. Yu, M. Jaggi, C. Musat, and M. Salzmann, "Evaluating the search phase of neural architecture search," in *Proc. Int. Conf. Learn. Representations*, 2020.

[25] A. Yang, P. M. Esperança, and F. M. Carlucci, "Nas evaluation is frustratingly hard," in *Proc. Int. Conf. Learn. Representations*, 2020.

[26] L. Xie *et al.*, "Weight-sharing neural architecture search: A battle to shrink the optimization gap," 2020, *arXiv:2008.01475*.

[27] Y. Liu *et al.*, "Search to distill: Pearls are everywhere but not the eyes," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 7539–7548.

[28] C. Li *et al.*, "Block-wisely supervised neural architecture search with knowledge distillation," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1989–1998.

[29] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," Citeseer, 2009.

[30] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[31] X. Dong and Y. Yang, "NAS-bench-201: Extending the scope of reproducible neural architecture search," in *Proc. Int. Conf. Learn. Representations*, 2020.

[32] A. Howard *et al.*, "Searching for mobilenetv3," in *Proc. Int. Conf. Comput. Vis.*, 2019, pp. 1314–1324.

[33] X. Dong and Y. Yang, "One-shot neural architecture search via self-evaluated template network," in *Proc. Int. Conf. Comput. Vis.*, 2019, pp. 3681–3690.

[34] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3/4, pp. 229–256, 1992.

[35] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 4095–4104.

[36] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 6105–6114.

[37] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "SegFormer: Simple and efficient design for semantic segmentation with transformers," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021.

[38] X. Chu, T. Zhou, B. Zhang, and J. Li, "Fair DARTS: Eliminating unfair advantages in differentiable architecture search," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 465–480.

[39] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[40] L. Shen, G. Sun, Q. Huang, S. Wang, Z. Lin, and E. Wu, "Multi-level discriminative dictionary learning with application to large scale image classification," *IEEE Trans. Image Process.*, vol. 24, no. 10, pp. 3109–3123, Oct. 2015.

[41] Y. Yang, J. Wu, H. Li, X. Li, T. Shen, and Z. Lin, "Dynamical system inspired adaptive time stepping controller for residual network families," *Assoc. Advance. Artif. Intell.*, vol. 34, no. 04, pp. 6648–6655, 2020.

[42] L. Liu *et al.*, "Deep learning for generic object detection: A survey," *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 261–318, 2020.

[43] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[44] G. Liu *et al.*, "Robust subspace segmentation by low-rank representation," in *Proc. Int. Conf. Mach. Learn.*, 2010, Art. no. 8.

[45] Y. Yang, H. Li, X. Li, Q. Zhao, J. Wu, and Z. Lin, "SOGNET: Scene overlap graph network for panoptic segmentation," *Assoc. Advance. Artif. Intell.*, vol. 34, no. 07, pp. 12 637–12 644, 2020.

[46] X. Li, Z. Zhong, J. Wu, Y. Yang, Z. Lin, and H. Liu, "Expectation-maximization attention networks for semantic segmentation," in *Proc. Int. Conf. Comput. Vis.*, 2019, pp. 9167–9176.

[47] T. Wang and H. Ling, "Gracker: A graph-based planar object tracker," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 6, pp. 1494–1501, Jun. 2018.

[48] W. Wang, J. Shen, and H. Ling, "A deep network solution for attention and aesthetics aware photo cropping," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 7, pp. 1531–1544, Jul. 2019.

[49] X. Li *et al.*, "Towards efficient scene understanding via squeeze reasoning," *IEEE Trans. Image Process.*, vol. 30, pp. 7050–7063, 2021.

[50] Z. Zhong, T. Shen, Y. Yang, Z. Lin, and C. Zhang, "Joint sub-bands learning with clique structures for wavelet domain super-resolution," *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.

[51] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2015.

[52] C. Cortes, X. Gonzalvo, V. Kuznetsov, M. Mohri, and S. Yang, "AdaNet: Adaptive structural learning of artificial neural networks," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 874–883.

[53] S. C. Smithson, G. Yang, W. J. Gross, and B. H. Meyer, "Neural networks designing neural networks: Multi-objective hyper-parameter optimization," in *Proc. 35th Int. Conf. Comput.-Aided Des.*, 2016, pp. 1–8.

[54] S. Saxena and J. Verbeek, "Convolutional neural fabrics," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 4053–4061.

[55] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2017.

[56] L. Xie and A. Yuille, "Genetic CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1379–1388.

[57] M. Tan *et al.*, "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2820–2828.

[58] J. Yu *et al.*, "BigNAS: Scaling up neural architecture search with big single-stage models," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 702–717.

[59] X. Dai *et al.*, "FBNetv3: Joint architecture-recipe search using neural acquisition function," 2020, *arXiv:2006.02049*.

[60] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.

[61] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once for all: Train one network and specialize it for efficient deployment," in *Proc. Int. Conf. Learn. Representations*, 2020.

[62] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[63] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.

[64] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7132–7141.

[65] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2423–2432.

[66] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," *Assoc. Advance. Artif. Intell.*, vol. 33, no. 01, pp. 4780–4789, 2019.

[67] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "NAS-bench-101: Towards reproducible neural architecture search," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 7105–7114.

[68] A. Zela, J. Siems, and F. Hutter, "NAS-bench-1shot1: Benchmarking and dissecting one-shot neural architecture search," in *Proc. Int. Conf. Learn. Representations*, 2020.

[69] B. Wang, Y. Sun, B. Xue, and M. Zhang, "Evolving deep convolutional neural networks by variable-length particle swarm optimization for image classification," in *Proc. IEEE Congr. Evol. Comput*, 2018, pp. 1–8.

[70] J. Liang, E. Meyerson, and R. Miikkulainen, "Evolutionary architecture search for deep multitask networks," in *Proc. Genet. Evol. Comput. Conf.*, 2018, pp. 466–473.

[71] Y. Sun, G. G. Yen, and Z. Yi, "Evolving unsupervised deep neural networks for learning meaningful representations," *IEEE Trans. Evol. Comput.*, vol. 23, no. 1, pp. 89–103, Feb. 2018.

[72] J.-D. Dong, A.-C. Cheng, D.-C. Juan, W. Wei, and M. Sun, "DPP-net: Device-aware progressive search for pareto-optimal neural architectures," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 517–531.

[73] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," in *Proc. Int. Conf. Learn. Representations*, 2018.

[74] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "SMASH: One-shot model architecture search through hypernetworks," in *Proc. Int. Conf. Learn. Representations*, 2018.

[75] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, "Understanding and simplifying one-shot architecture search," in *Proc. IEEE Int. Conf. Learn. Representations*, 2018, pp. 550–559.

[76] B. Wu *et al.*, "FBNet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10734–10742.

[77] Z. Guo *et al.*, "Single path one-shot neural architecture search with uniform sampling," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 544–560.

[78] X. Chu, B. Zhang, and R. Xu, "MoGA: Searching beyond mobilenetv3," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2020, pp. 4042–4046.

[79] X. Chu, B. Zhang, R. Xu, and J. Li, "FairNAS: Rethinking evaluation fairness of weight sharing neural architecture search," in *Proc. Int. Conf. Comput. Vis.*, 2019, pp. 12 239–12 248.

[80] X. Li, C. Lin, C. Li, M. Sun, W. Wu, J. Yan, and W. Ouyang, "Improving one-shot NAS by suppressing the posterior fading," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 13836–13845.

[81] Y. Xu *et al.*, "PC-DARTS: Partial channel connections for memory-efficient architecture search," in *Proc. Int. Conf. Learn. Representations*, 2020.

[82] Y. Yang, H. Li, S. You, F. Wang, C. Qian, and Z. Lin, "ISTA-NAS: Efficient and consistent neural architecture search by sparse coding," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020.

[83] R. Istrate, F. Scheidegger, G. Mariani, D. Nikolopoulos, C. Bekas, and A. C. I. Malossi, "TAPAS: Train-less accuracy predictor for architecture search," *Assoc. Advance. Artif. Intell.*, vol. 33, no. 01, pp. 3927–3934, 2019.

[84] C. Wong, N. Houlsby, Y. Lu, and A. Gesmundo, "Transfer learning with neural automl," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018.

[85] M. Wistuba, "XferNAS: Transfer neural architecture search," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2020, pp. 247–262.

[86] J. Peng, M. Sun, Z. Zhang, T. Tan, and J. Yan, "Efficient neural architecture transformation searchin channel-level for object detection," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020.

[87] J. Fang *et al.*, "FNA++: Fast network adaptation via parameter remapping and architecture search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 9, pp. 2990–3004, Sep. 2021.

[88] Z. Lu, G. Sreekumar, E. Goodman, W. Banzhaf, K. Deb, and V. N. Boddeti, "Neural architecture transfer," *IEEE Trans. Pattern Anal. Mach. Intell*, vol. 43, no. 9, pp. 2971–2889, Sep. 2021.

[89] H. Peng, H. Du, H. Yu, Q. Li, J. Liao, and J. Fu, "Cream of the crop: Distilling prioritized paths for one-shot neural architecture search," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020.

[90] H. Pham, Z. Dai, Q. Xie, and Q. V. Le, "Meta pseudo labels," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 11557–11568.

[91] C. Sciuto, K. Yu, M. Jaggi, C. Musat, and M. Salzmann, "Evaluating the search phase of neural architecture search," in *Proc. Int. Conf. Learn. Representations*, 2019.

[92] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2012, vol. 25, pp. 1097–1105.

[93] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.

[94] A. Paszke *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 8026–8037.

[95] X. Dong, L. Liu, K. Musial, and B. Gabrys, "NATS-Bench: Benchmarking NAS algorithms for architecture topology and size," *IEEE Trans. Pattern Anal. Mach. Intell*, early access, Jan. 26, 2021, doi: 10.1109/TPAMI.2021.3054824.

[96] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, no. Feb, pp. 281–305, 2012.

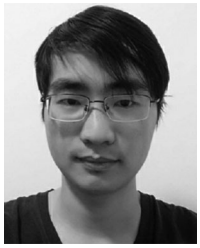
[97] S. Falkner, A. Klein, and F. Hutter, "BOHB: Robust and efficient hyperparameter optimization at scale," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1437–1446.

[98] P. Chrabaszcz, I. Loshchilov, and F. Hutter, "A downsampled variant of imagenet as an alternative to the cifar datasets," 2017, *arXiv:1707.08819*.

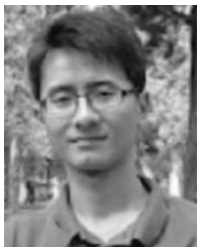
[99] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proc. Brit. Mach. Vis. Conf.* 2016, pp. 87.1–87.12.

[100] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1492–1500.

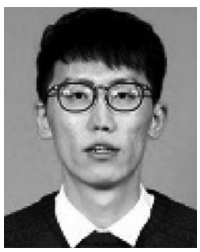
[101] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, "Neural architecture optimization," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018.
[102] Y. Xu *et al.*, "Latency-aware differentiable neural architecture search," 2020, *arXiv:2001.06392*.
[103] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," in *Proc. Int. Conf. Learn. Representations*, 2017.
[104] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, vol. 22, pp. 400–407, 1951.
[105] G. Larsson, M. Maire, and G. Shakhnarovich, "FractalNet: Ultra-deep neural networks without residuals," in *Proc. Int. Conf. Learn. Representations*, 2017.
[106] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations*, 2015.
[107] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," 2017, *arXiv:1708.04552*.
[108] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.
[109] A. Gholami *et al.*, "SqueezeNext: Hardware-aware neural network design," in *Proc. Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 1638–1647.
[110] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 116–131.
[111] N. Nayman, A. Noy, T. Ridnik, I. Friedman, R. Jin, and L. Zelnik, "XNAS: Neural architecture search with expert advice," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019.
[112] A. Noy *et al.*, "ASAP: Architecture search, anneal and prune," 2019, *arXiv:1904.04123*.
[113] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
[114] P. Goyal *et al.*, "Accurate, large minibatch SGD: Training imagenet in 1 hour," 2017, *arXiv:1706.02677*.
[115] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2818–2826.
[116] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6848–6856.
[117] S. Hu *et al.*, "DSNAS: Direct neural architecture search without parameter retraining," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 12 084–12 092.
[118] M. Tan and Q. V. Le, "MixConv: Mixed depthwise convolutional kernels," in *Proc. Brit. Mach. Vis. Conf.*, 2019.
[119] X. Chu, B. Zhang, Q. Li, R. Xu, and X. Li, "SCARLET-NAS: Bridging the gap between stability and scalability in weight-sharing neural architecture search," in *Proc. Int. Conf. Comput. Vis.*, pp. 317–325, 2021.
[120] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "AutoAugment: Learning augmentation strategies from data," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 113–123.
[121] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz , "mixup: Beyond empirical risk minimization," in *Proc. Int. Conf. Learn. Representations*, 2018.
[122] X. Chu, X. Li, Y. Lu, B. Zhang, and J. Li, "MixPath: A unified approach for one-shot neural architecture search," 2020, *arXiv:2001.05887*.
[123] S. Mehta, M. Rastegari, A. Caspi, L. Shapiro, and H. Hajishirzi, "ESPNet: Efficient spatial pyramid of dilated convolutions for semantic segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 552–568.
[124] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "ICNet for real-time semantic segmentation on high-resolution images," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 405–420.
[125] J. Wang *et al.*, "Deep high-resolution representation learning for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 10, pp. 3349–3364, Oct. 2019.
[126] Y. Zhang, Z. Qiu, J. Liu, T. Yao, D. Liu, and T. Mei, "Customizable architecture search for semantic segmentation," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 11 641–11 650.
[127] A. Howard *et al.*, "Searching for mobileNetV3," in *Proc. Int. Conf. Comput. Vis.*, 2019, pp. 1314–1324.
[128] W. Chen, X. Gong, X. Liu, Q. Zhang, Y. Li, and Z. Wang, "FasterSeg: Searching for faster real-time semantic segmentation," in *Proc. Int. Conf. Learn. Representations*, 2020.
[129] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang, "BiSeNet V2: Bilateral network with guided aggregation for real-time semantic segmentation," *Int. J. Comput. Vis.*, vol. 129, pp. 3051–3068, 2021.
[130] A. Shaw, D. Hunter, F. Landola, and S. Sidhu, "SqueezeNAS: Fast neural architecture search for faster semantic segmentation," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, 2019.
[131] M. Orsic, I. Kreso, P. Bevandic, and S. Segvic, "In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 12607–12616.
[132] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. Int. Conf. Comput. Vis.*, 2017, pp. 2980–2988.
[133] T.-Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.
[134] K. Chen *et al.*, "MMDetection: Open mmlab detection toolbox and benchmark," 2019, *arXiv:1906.07155*.
[135] X. Li, Y. Yang, Q. Zhao, T. Shen, Z. Lin, and H. Liu, "Spatial pyramid based graph reasoning for semantic segmentation," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 8950–8959.
[136] M. Cordts *et al.*, "The cityscapes dataset for semantic urban scene understanding," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3213–3223.
[137] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ADE20K dataset," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 633–641.
[138] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3431–3440.
[139] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2881–2890.
[140] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 801–818.



Hongyuan Yu received the BSc degree from the Nankai University (NKU), Tianjin, China, in 2017. He is currently working toward the PhD degree with the Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing, China, and the AI School, University of Chinese Academy of Sciences (UCAS), Beijing, China. His research interests include machine learning and pattern recognition.

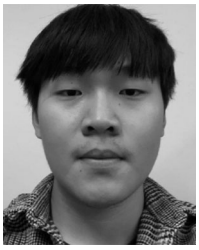


Houwen Peng received the PhD degree from the NLPR, Institution of Automation, Chinese Academy of Sciences in 2016. He is currently a senior researcher working on computer vision and deep learning with Microsoft Research as of 2018. Before that he was a senior engineer with Qualcomm AI Research. From 2015 to 2016, he worked as a visiting research scholar with Temple University. His research interests include tiny and efficient deep learning, video object tracking, segmentation and detection, vision transformer, neural architecture search, model compression, vision-language intelligence, saliency detection, etc. He has served as the area chairs /senior program committee member for ACM Multimedia and AAAI.

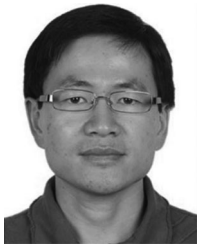


Yan Huang received the PhD degree from the AI School, University of Chinese Academy of Sciences (UCAS), in 2017. Since July 2017, He has joined the National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences (CASIA) as an assistant professor. His research interests include machine learning and pattern recognition. He has published papers in the leading international journals and conferences such as the *IEEE Transactions on Pattern Analysis and Machine Intelligence*, the *IEEE Transactions on Image Processing*, NeurIPS, CVPR, ICCV and ECCV.

Jianlong Fu received the PhD degree in pattern recognition and intelligent system from the Institute of Automation, Chinese Academy of Science, in 2015. He is currently a lead researcher with the Multimedia Search and Mining Group, Microsoft Research Asia (MSRA). He has authored or coauthored more than 40 papers in journals and conferences, and one book chapter. His current research interests include computer vision, computational photography, vision, and language. He received the Best Paper Award from ACM Multimedia 2018, and has shipped core technologies to a number of Microsoft products, including Windows, Office, Bing Multimedia Search, Azure Media Service, and XiaoIce. He is an area chair of ACM Multimedia 2018, ICME 2019. He serves as a lead organizer and a guest editor for *IEEE Transactions on Pattern Analysis and Machine Intelligence* Special Issue on Fine-grained Categorization.



Hao Du is currently working toward the PhD degree with the City University of Hong Kong (CityU), Hong Kong, China. His research interests include neural architecture design and search and video object tracking.



Liang Wang received the BEng and MEng degrees from Anhui University, in 1997 and 2000, respectively, and the PhD degree from the Institute of Automation, Chinese Academy of Sciences (CASIA), in 2004. From 2004 to 2010, he was a research assistant with Imperial College London, United Kingdom, and Monash University, Australia, a research fellow with the University of Melbourne, Australia, and a lecturer with the University of Bath, United Kingdom, respectively. Currently, he is a full professor of the Hundred Talents Program with the National Lab of Pattern Recognition, CASIA. His major research interests include machine learning, pattern recognition, and computer vision. He has widely published in highly ranked international journals such as *IEEE Transactions on Pattern Analysis and Machine Intelligence* and *IEEE Transactions on Image Processing*, and leading international conferences such as CVPR, ICCV, and ECCV. He is a fellow of the IAPR.

Haibin Ling received the BS and MS degrees from Peking University in 1997 and 2000, respectively, and the PhD degree from the University of Maryland, College Park, in 2006. From 2000 to 2001, he was an assistant researcher with Microsoft Research Asia. From 2006 to 2007, he worked as a postdoctoral scientist with the University of California Los Angeles. In 2007, he joined Siemens Corporate Research as a research scientist; then, from 2008 to 2019, he worked as a faculty member with the Department of Computer Sciences, Temple University. In fall 2019, he joined Stony Brook University as a SUNY empire innovation professor with the Department of Computer Science. His research interests include computer vision, augmented reality, medical image analysis, and human computer interaction. He received Best Student Paper Award for ACM UIST (2003), NSF CAREER Award (2014), Yahoo Faculty Research Award (2019), and Amazon AWS Machine Learning Research Award (2019). He serves as associate editors for several journals including *IEEE Transactions on Pattern Analysis and Machine Intelligence* (PAMI), *Pattern Recognition* (PR), and *Computer Vision and Image Understanding* (CVIU), and has served as aarea chairs various times for CVPR and ECCV.