PP-NAS: Searching for Plug-and-Play Blocks on Convolutional Neural Networks

Anqi Xiao, Biluo Shen, Jie Tian, *Fellow, IEEE*, and Zhenhua Hu, *Senior Member, IEEE*

***Abstract*— Multiscale features are of great importance in modern convolutional neural networks, showing consistent performance gains on numerous vision tasks. Therefore, many plug-and-play blocks are introduced to upgrade existing convolutional neural networks for stronger multiscale representation ability. However, the design of plug-and-play blocks is getting more and more complex, and these manually designed blocks are not optimal. In this work, we propose PP-NAS to develop plug-and-play blocks based on neural architecture search (NAS). Specifically, we design a new search space PPConv and develop a search algorithm consisting of one-level optimization, zero-one loss, and connection existence loss. PP-NAS minimizes the optimization gap between super-net and subarchitectures and can achieve good performance even without retraining. Extensive experiments on image classification, object detection, and semantic segmentation verify the superiority of PP-NAS over state-of-the-art CNNs (e.g., ResNet, ResNeXt, and Res2Net). Our code is available at https://github.com/ainieli/PP-NAS.**

***Index Terms*— Multiscale, neural architecture search (NAS), plug-and-play, representation learning.**

I. Introduction

MULTISCALE features are important for visual tasks in natural scenes. Objects within a single image may have various sizes and the same object may have different sizes between multiple images. Then, different parts of an object are usually of different sizes and may be helpful for understanding the object. Furthermore, under some circumstances, recognition of an object may be difficult from the object itself, but much easier when relying on essential context information from multiscale features. Thus, it is of great importance to capture multiscale features that benefit computer vision tasks, including image classification [1], [2], [3], [4], [5], object detection [6], [7], [8], [9], [10], instance segmentation [11], [12], semantic segmentation [13], [14], keypoint detection [15], salient object detection [16], [17], medical image analysis [18], [19], [20], [21], [22], and image deraining [23].

The key to capturing multiscale features in convolutional neural networks is the design of network architecture. Many plug-and-play blocks [1], [2], [3], [4], which can be easily integrated into existing networks by replacing the regular convolution operation, were proposed to improve the multiscale representation ability of the networks. However, the design of these blocks has become more and more complex and requires significant architecture engineering. Moreover, these manually designed blocks may contain human bias and are not optimal. We believe that a new design pattern would be introduced to facilitate the development of plug-and-play blocks.

To address these problems, we advocate for the use of neural architecture search (NAS) to find better plug-and-play blocks automatically. In particular, a new search space for plug-and-play blocks is proposed. Following the design of previous works, the new search space can be easily integrated into existing networks by only replacing the main convolution operation. Therefore, we name the new search space as plug-and-play convolution (PPConv). To better focus on the design of plug-and-play blocks, we simply choose the most widely used ResNet with bottleneck structure as the macro architecture and replace the 3×3 convolutions with the searchable plug-and-play blocks. The concrete operations and connections inside the plug-and-play blocks will be searched and derived.

After the definition of search space, differentiable NAS (DNAS or DARTS) [24], [25], [26], [27], [28] methods are used to jointly optimize the network weights and architecture parameters in a weight-sharing super-net via gradient descent. The final architecture is derived from the trained super-net at the end of the search phase. Unlike most DARTS-based methods using bilevel optimization which suffers from a heavy computational burden and inaccurate estimation of architectural gradients, we apply one-level optimization to speed up and simplify the search procedure. We also add strong

Manuscript received 16 October 2021; revised 13 December 2022; accepted 27 March 2023. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 92059207, Grant 62027901, Grant 81930053, and Grant 81227901; in part by the Chinese Academy of Sciences (CAS) Youth Interdisciplinary Team under Grant JCTD-2021-08; in part by the Zhuhai High-Level Health Personnel Team Project under Grant Zhuhai HLHPTP201703; and in part by the Cloud Tensor Processing Unit (TPUs) from Google's TPU Research Cloud (TRC). *(Anqi Xiao and Biluo Shen contributed equally to this work.) (Corresponding authors: Jie Tian; Zhenhua Hu.)*

Anqi Xiao, Biluo Shen, and Zhenhua Hu are with the CAS Key Laboratory of Molecular Imaging, Beijing Key Laboratory of Molecular Imaging, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 101408, China (e-mail: xiaoanqi2020@ia.ac.cn; shenbiluo2019@ia.ac.cn; zhenhua.hu@ia.ac.cn).

Jie Tian is with the CAS Key Laboratory of Molecular Imaging, Beijing Key Laboratory of Molecular Imaging, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 101408, China, also with the Beijing Advanced Innovation Center for Big Data-Based Precision Medicine, School of Engineering Medicine, Beihang University, Beijing 100191, China, and also with the Engineering Research Center of Molecular and Neuro Imaging of Ministry of Education, School of Life Science and Technology, Xidian University, Xi'an 710071, China (e-mail: tian@ieee.org).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3264551.

Digital Object Identifier 10.1109/TNNLS.2023.3264551

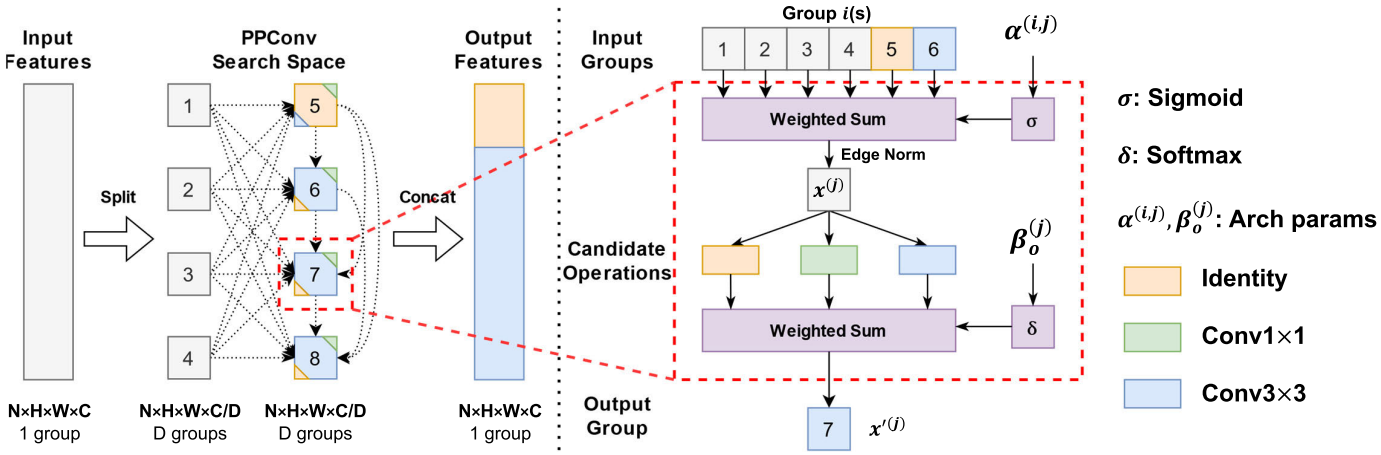

Fig. 1. Pipeline of PP-NAS-induced PPConv. (Left) Process of PPConv to calculate input features with shape $N \times H \times W \times C$. The input features are split into D groups for search, where $D = 4$ in the example. The middle part shows the overview of the search space. Dotted lines are candidate connections to pass the information flow. Squares with triangles in the corner show candidate operations for the search, where the center color indicates the dominant operation with the largest weight. The output features of each group are marked with the same color as the dominant operation. (Right) Details of the process in the output groups of the red dashed box on the left with $j = 7$. The forward process contains a sum of input features weighted by connection parameters α, followed by a sum of candidate operations processed results weighted by operation parameters β. Architecture parameters α and β are learnable and determine the derived architecture.

regularization to prevent the failure caused by overfitting [29], [30]. We summarize our main contributions as follows.

1) We propose a new search space PPConv for plug-and-play blocks and integrate it into ResNets to get PP-ResNets, which is different from most previous NAS works based on NASNet or MobileNet search space.

2) We introduce a search algorithm composed of one-level optimization, zero-one loss, and connection existence loss. It can significantly outperform random search and find good connections and operations in the new search space.

3) PP-NAS minimizes the optimization gap between super-net and derived architectures, which achieves good performance even without retraining.

4) PP-ResNets outperform state-of-the-art ResNet-style CNNs and show consistent gains on datasets and benchmarks, including Canadian Institute For Advanced Research (CIFAR), ImageNet, Visual Object Classes (VOC), Common Objects in COntext (COCO), and Cityscapes.

Our work also has a conference version [31]. Compared with it, additional contents are highlighted, including a figure for the pipeline of PPConv (Fig. 1), comparison with other NAS methods in classification (Tables I and III), random search (Table II), detailed object detection results for each class (Table V), visualization of semantic segmentation results (Figs. 6 and 7), latency experiments (Table VII), ablation studies for loss terms (Table VIII), and discussions about PP-NAS (Section V and Fig. 5).

II. Related Work

A. Multiscale Network and Plug-and-Play Block

The key to capturing multiscale features in convolutional neural networks is the design of network architecture. Res2Net [1] proposed to connect split small filter groups in a hierarchical residual-like style to increase multiscale representation strength. MixConv [2] mixed up multiple kernel sizes in a single depthwise convolution to capture patterns at different resolutions for better accuracy and efficiency. PyConv [3] introduced pyramidal convolution that contains a pyramid of kernels with varying size and depth to capture different levels of details in the scene. HS-ResNet [4] addressed the hierarchical-split block that contains many hierarchical splits and concatenates connections within a single residual block.

B. Differentiable Neural Architecture Search

DNAS is one of the one-shot search methods, in which an overparameterized super-net containing all candidate architectures is trained only once. DARTS [24] introduced a differentiable framework by relaxing the search space so that the architecture parameters could be differentiable and optimized together with the network weights by gradient descent. Despite its simplicity, many follow-up works revealed some of its drawbacks, such as instability [32], the inevitable aggregation of skip connections [29], and the gap between the search and the evaluation [33]. P-DARTS [34] designed a progressive search strategy to bridge the depth gap between super-net and subarchitectures. FairDARTS [35] proposed a zero-one loss combined with sigmoid function to alleviate the issue of discretization discrepancy. RobustDARTS [29] showed that adding regularization strategies can robustify DARTS to find solutions with less curvature and better generalization performance. GOLD-NAS [30] enlarged the search space to contain more than 10^{160} candidates and used one-level optimization with a variable resource constraint to explore this large search space.

III. Methodology

A. Search Space

The pipeline of the proposed PP-NAS-induced PPConv is shown in Fig. 1. To make it easier to understand, we also compare the structure of group convolutions [36] and the blocks used by Res2Net with PPConv in Fig. 2. Group convolutions split the features into D groups, apply convolution on each group, and concatenate all output feature groups. The Res2Net

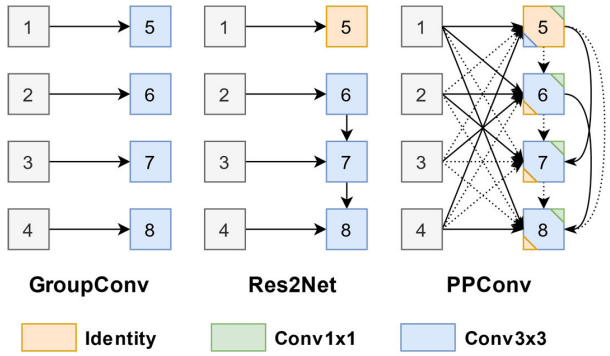

Fig. 2. Comparison between group convolution (GroupConv), the blocks used by Res2Net, and our proposed searchable PPConv. The solid lines represent the selected connections, and the dotted lines represent the pruned connections. The selected operation is placed at the center of the output feature group and the pruned operations are in the corner. The candidate operations include identity, 1×1 convolution, and 3×3 convolution. Vertical lines and curved lines on the right indicate that feature groups with smaller indexes can be part of the input of feature groups with larger indexes.

blocks add connections between output feature groups to increase the actual network depth. Besides, the convolution operation applied to the first group is replaced as an identity operation. This design reduces parameters or, in another way, allows wider convolution for the rest of the groups. It also encourages feature reuse, which follows DenseNet [37].

PPConv is a generalization of group convolution and Res2Net blocks. Let us first split the input features into D groups evenly and denote them as $x^{(i)}(0 < i \leq D)$; then, we also have D groups of intermediate features denoted as $x^{(i)}(D < i \leq 2D)$ and D groups of output feature maps denoted as $x'^{(i)}(D < i \leq 2D)$. A group of features will be called a node for convenience in the following. The connection between $x^{(i)}$ and $x^{(j)}$ is a directed edge (i, j), where $0 < i < j$ and $D < j \leq 2D$. Each intermediate node is computed based on all of its predecessors, the weight of which is associated with connection parameters α. Each output node is the output of the associated operation applied on the intermediate node, the weight of which is associated with operation parameters β. α and β are architecture parameters to learn and will be described in the next part. With this design of search space, learning the block is reduced to learning the connections between nodes and the operations applied on intermediate nodes. It should be noted that their learning is decoupled, which differs from previous works.

The candidate operations include identity, 1×1 convolution, and 3×3 convolution. By including these operations, we ensure that group convolution and Res2Net blocks are also in the search space. Zero operation [24] is not included as candidates because we want to fix the number of output nodes and the number of output features to the same as the input.

Next, we describe the macro architecture. PPConv is designed to replace the main convolution (usually 3×3 convolution) in networks. Therefore, it is easy to integrate it into existing mainstream network architectures. To better focus on the design of new plug-and-play blocks, we simply choose the most widely used ResNet as the macro architecture. Specifically, ResNet with bottleneck structure is used and the middle 3×3 convolution with stride 1 in the bottleneck is replaced with PPConv for all of the following experiments. We call ResNet with PPConv as PP-ResNet.

For datasets with different resolutions (CIFAR and ImageNet), ResNet architectures with an appropriate number of stages and times of downsampling are required. Therefore, the concrete architecture depends on the dataset, following the design of the original ResNet [38]. For example, ResNets on CIFAR have three stages and are downsampled two times, and ResNets on ImageNet have four stages and are downsampled five times. PPConv structures are searched separately across different stages and shared between blocks in the same stage.

B. Differentiable Architecture Search

Following previous works, we use continuous relaxation to make the search space continuous and search procedure differentiable. Every edge (i, j) between nodes $x^{(i)}$ and $x^{(j)}$ is parameterized by $\alpha^{(i,j)}$. As we want to have a variable number of connections for each node rather than a fixed number of connections, softmax activation that encourages competition between edges is not appropriate. Instead, sigmoid activation (σ) is used, and the edges of a node will cooperate with each other, leading to smoother information flow. More importantly, each edge will be switched on or off independently according to its corresponding connection parameter. Formally, the intermediate node is computed as follows:

$$x^{(j)} = \frac{\sum_{i<j} \sigma\left(\alpha^{(i,j)}\right) x^{(i)}}{\sum_{i<j} \sigma\left(\alpha^{(i,j)}\right)}. \tag{1}$$

Note that the output is normalized by the sum of all associated $\sigma(\alpha)$ because intermediate nodes have a different number of predecessors and the norm of nodes will differ significantly without normalization, which might hurt model stability.

Next, we will discuss how to search for operations. Let O denote a set of possible operations (identity, 1×1 convolution, and 3×3 convolution in this article) where every operation is to be applied to the intermediate nodes to get output nodes. The categorical choice of a particular operation for $x'^{(j)}$ is relaxed to a softmax over all candidates

$$\bar{o}^{(j)}(x) = \sum_{o \in O} \frac{\exp\left(\beta_o^{(j)}\right)}{\sum_{o' \in O} \exp\left(\beta_{o'}^{(j)}\right)} o(x). \tag{2}$$

The operation weights for a node are parameterized by a vector $\beta^{(j)}$ of dimension $|O|$. At the end of search phase, the mixed operation $\bar{o}^{(j)}$ can be replaced with the most likely operation, i.e., $o^{(j)} = \text{argmax}_{o \in O} \beta_o^{(j)}$.

After relaxation, we jointly optimize the architecture parameters α, β, and the network weights w. Previous differentiable methods recognized it as a bilevel optimization problem and alternatively optimized the training loss on the training set and validation loss on the validation set using gradient descent. However, as pointed out in [30], bilevel optimization suffers from a heavy computational burden and inaccurate estimation of architecture gradients. Instead, we use one-level

optimization that optimizes the architecture α and β and the weights w at the same time while only on the training set. To avoid the failure of one-level optimization which might be caused by overfitting, we also add strong data augmentation and regularization. With one-level optimization, the search procedure is exactly the same as training a classification network, except for the use of two different optimizers for the architecture parameters α and β and the model weights correspondingly.

C. Architecture Derivation and Optimization Gap

At the end of the search, when the architecture parameters α and β are fully optimized, we derive concrete connections and operations to form discrete architectures. For connections between nodes, we select all possible choices if their associated α is above $\sigma_{\text{threshold}}$. As for operations, we simply choose the one with the largest weight β. The intermediate nodes are computed based on all of their predecessors with the same weight: $x^{(j)} = \sum_{i<j} x^{(j)}/(j-1)$, $D < j \leq 2D$.

As is well known, the biggest pitfall of weight-sharing methods is the optimization gap between the super-net and the subarchitectures [39]. Without extra constraints, at the end of search, the final $\sigma(\alpha)$ will be only slightly larger or less than 0.5, thus resulting in the optimization gap. To alleviate this gap, an extra zero-one loss [35], [40] is explicitly added to push the sigmoid value of architecture parameters toward 0 or 1, formally as follows:

$$L_{0-1} = -\sum_{i,j} \left(\sigma\left(\alpha^{(i,j)}\right) - 0.5\right)^2. \tag{3}$$

After this, it is also possible that no connection is selected from the input nodes or to the intermediate nodes if their associated $\sigma(\alpha)$ are all below $\sigma_{\text{threshold}}$. If any of the input nodes have no connection to the intermediate nodes, this group of feature maps is completely dropped, and a part of the input information will be lost. On the other hand, if any of the intermediate nodes have no connection from the other nodes, the number of output nodes will change, which is not expected.

To solve these problems, we coerce another connection existence loss to ensure the existence of at least one connection. Meanwhile, we also introduce max operation to avoid unnecessary constraints on architecture learning when the connections already exist for each feature group. With this design, the loss will not make the derived architectures sparse. To achieve the expectations, the loss is formulated as follows:

$$L_{\text{conn}} = \sum_{0<i\leq D} \max\left(1 - \sum_{D<j\leq 2D} \sigma\left(\alpha^{(i,j)}\right), 0\right)^2 + \sum_{D<j\leq 2D} \max\left(1 - \sum_{0<i<j} \sigma\left(\alpha^{(i,j)}\right), 0\right)^2. \tag{4}$$

The square is applied to ensure that the gradient of each connection parameter relates to other connection parameters, which is consistent with the characteristic of one-shot NAS that has mutual influence of different subarchitectures. However, the loss without a square may work as well. As the zero-one loss pushes $\sigma(\alpha)$ toward 0 or 1, the connection loss should work well.

In conclusion, the total loss to optimize is the sum of the classification loss (cross entropy), the $L2$ loss for all learnable parameters, the zero-one loss, and the connection existence loss for α formally as follows:

$$L = L_{\text{CE}} + L_{L2} + L_{0-1} + L_{\text{conn}}. \tag{5}$$

Weights for the four parts of losses may be assigned and tuned, but we omit them here for simplicity. The performance may be further improved if carefully tuning the weights.

Our goal is to bridge the optimization gap and derive discrete architectures from the super-net with less discretization error. To achieve this goal, we adopt one-level optimization that jointly updates model weights and architecture parameters. The optimization process is written as follows:

$$\min_{w,\alpha,\beta} L(w, \alpha, \beta) \tag{6}$$

where w indicates model weights. One-level optimization allows the parameters to be optimized directly from the target without approximation of the optimized model weight that is used to reduce search cost, which is different from bilevel optimization and reduces the optimization gap.

Besides, the design of loss functions helps to eliminate the optimization gap as well, where L_{0-1} tries to keep the forward result of the derived architecture similar to the super-net in one PPConv. However, we should admit that the deviation between the forward results of the derived architecture and super-net will accumulate with the increase of depth and complexity of PPConv, which is difficult to eliminate. In addition, the depth and width gap still exist. We notice that PPConv decouples the search of connections and operations, leading to much less usage of memory and searching cost. Then, it is possible to use precisely the same depth and width for the architecture search as the architecture evaluation. As a result, we can directly obtain a trained network by pruning unimportant connections and operations at the end of the search, with a very small performance drop.

IV. Experimental Results

A. Searching on CIFAR

CIFAR-10 [41] is a standard image classification dataset consisting of 50 k training images with 5 k images per class and 10 k testing images with 1 k images per class. The resolution of each image is 32 × 32. CIFAR-100 is just like CIFAR-10 and has the same number of images as CIFAR-10 but with 100 fine-grained classes. The training set of CIFAR-100 has 50 k images with 500 images per class, and the test set has 10 k images with 100 images per class. Since we use one-level optimization, there is no need to split the training set for another validation set, so we conduct the architecture search on CIFAR-10/100 with all the training images.

We use ResNet-110 with the bottleneck structure as the backbone network for CIFAR-10/100. To achieve comparable performance with Wide ResNet (WRN) [42], some improvements are applied. In the original paper, ResNet-110 is based on the basic blocks that are different from ours, so every

stage in our ResNet-110 has 12 residual blocks. In addition, the number of channels of three stages of ResNet-110 in the original paper is 16, 32, and 64, but we set them as 64, 128, and 256. The resulting ResNet-110 has a total of 18.5 M parameters. Another tweak to the architecture is adding a 3×3 average pooling layer with a stride of 2 before the convolution with a stride changed to 1. This tweak is also applied to the ResNeXt and Res2Net for a fair comparison.

As mentioned before, to minimize the optimization gap, we try to keep most of the network and hyperparameter settings the same between the search and the evaluation. The model weights w are optimized by SGD with an initial learning rate of 0.1, a momentum of 0.9, and a weight decay of 5e$-$4, and the architecture parameters α and β are optimized by Adam [43], with a learning rate of 1e$-$3, a momentum (0.9, 0.999), and a batch size of 128. As we use explicit L2 loss for the architecture parameters, no weight decay is used for Adam. The super-net is trained for 300 epochs with additional data augmentation, including Cutout [44] and AutoAugment [45]. The learning rate for SGD is adjusted according to a cosine schedule [46] and the learning rate for Adam is fixed. We set the weights for L_{CE}, L_{0-1}, and L_{conn} in the loss of (5) to 1.0 for simplicity, while the weight for L_{L2} is the same as the weight decay for SGD. The initial value for the connection parameters is set to 0. Architecture parameters of each stage are shared to reduce the calculation. After training the super-net, $\sigma_{\text{threshold}}$ is set to 0.9 to prune the unimportant connections.

In the evaluation, the network depth and width are not changed, and the architecture is derived from the super-net. For CIFAR-10/100, we have two training settings for a fair comparison between other works: standard and augmented. Under the standard setting, we train the network for 200 epochs with only random crop, random horizontal flip, and normalization as the data augmentation. For the augmented setting, the network is trained for 600 epochs with Cutout, AutoAugment, and Mixup [47]. The cutout length is 16 and the mixup ratio α_{mix} is 0.2. Other hyperparameters are precisely the same as in the search phase. For each type of network architecture, we repeat the evaluation five times under the standard setting and three times under the augmented setting and report the mean of these results.

For a fair comparison, ResNet, ResNeXt, and Res2Net are reimplemented and trained under the same training settings as PP-ResNet. ResNeXt has a cardinality of 4 and the number of channels for each group is 24, formally as $4c \times 24w$. Res2Net has a split of 4 and the number of channels of each split is 26, formally as $26w \times 4s$, which is the same as PP-ResNet.

To ensure that PP-ResNet has a similar number of parameters with ResNeXt and Res2Net, we keep exactly one "identity" operation after searching. This decision slightly narrows search space and may affect the final performance. Similarly, many previous works of DARTS [24], [48] limited the number of "skip connect" operations to 1 or 2.

Besides, to show the relative performance of PP-ResNet compared with other NAS methods, we also compare the performance of PP-ResNet with some related differentiable NAS methods. However, we should admit that the comparison is not fair due to the divergent design of search space. These methods are shown as references to the performance improvement that can help readers better understand the effectiveness of PP-ResNet compared with manually designed ResNets.

TABLE I
RESULTS OF DIFFERENT ARCHITECTURES ON CIFAR-10/100

Architecture	Test Error (%)		Params
	C10	C100	(M)
DARTS (2nd) [24]	2.76	-	3.3
FairDARTS [35]	2.54	-	3.3
GOLD-NAS [30]	2.53	-	3.7
CyDAS [49]	2.48	15.69	3.9
SENet + Shake-Shake†	2.12	13.81	26.2
PyramidNet272 + Shakedrop†	1.70	11.70	26.0
ResNet-110	3.77	18.13	18.1
ResNeXt-110 (4c×24w)	3.71	17.73	18.1
Res2Net-110 (26w×4s)	3.67	17.46	18.5
PP-ResNet-110 (26w×4s)	3.50	17.33	18.5
PP-ResNet-110* (26w×4s)	1.94	13.67	18.5

† *Trained for 1800 epochs.*
* *Trained under the augmented setting (600 epochs).*

In Table I, we compare the test error and the number of parameters of the discovered architectures with other networks. Our PP-ResNet has an improvement of 0.17% over Res2Net on CIFAR-10 and 0.13% on CIFAR-100 under the standard setting. We also find that PP-ResNet can outperform the comparing differentiable methods when using the augmented setting, while it obviously performs worse than them when using the basic setting. The results demonstrate the potential of PP-NAS for integration with other tricks to reach better performance.

The architectures discovered on CIFAR-10/100 are shown in Fig. 3. We find that there is no 1×1 convolution in the discovered architectures. According to many related works of NAS, we speculate that the nonparametric identity operation has some special advantages over the parametric convolution operations during optimization. Meanwhile, between the parametric operations, 3×3 convolution has $8\times$ more parameters than 1×1 convolution, resulting in a significant advantage. We plan to explain it from more perspectives in the future.

B. Comparison With Random Search

Random search is a competitive baseline for hyperparameter optimization [50] and NAS [51]. Many recent works prove that random search with early stopping or other heuristics performs similarly with NAS methods. PP-NAS has a relatively small search space compared with many NAS works, which has the potential to suffer from degenerated architecture after

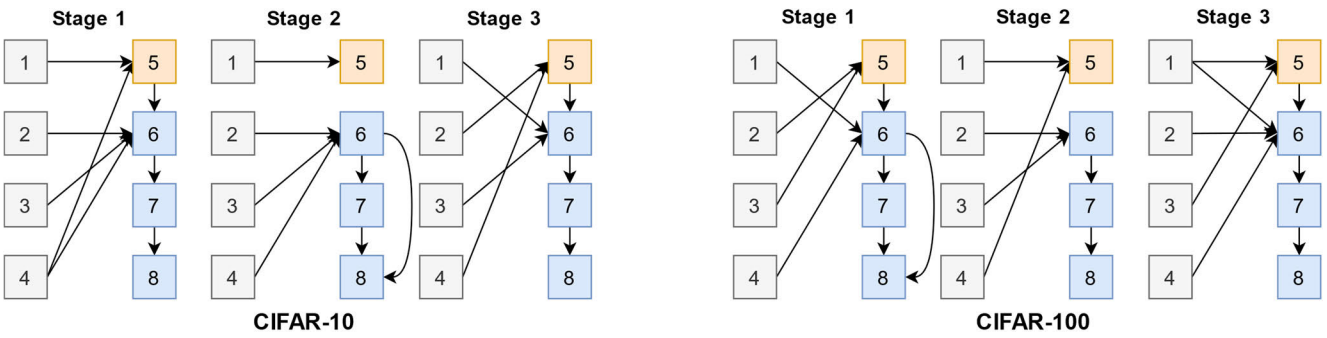

Fig. 3. Discovered blocks on CIFAR-10/100 for the three different stages of PP-ResNet-110.

TABLE II
COMPARISON OF RANDOM SEARCH ON CIFAR-10/100

Architecture	Test Error (%)	
	C10	C100
Random Search (Full)	3.74	17.73
Random Search (Constrained)	3.69	17.69
PP-ResNet-110 (26w×4s)	**3.50**	**17.33**

searching, or the architectures in the search space show similar performances. To evaluate the effectiveness of the proposed search algorithm, we conduct random search experiments on CIFAR-10/100 and compare them with our PP-NAS search results.

Considering the design of the proposed search space, we conduct two types of random searches: full random search and constrained random search. For full random search, we first initialize the architecture parameters and then derive architectures from them directly without optimization. It is possible that no connection is selected from the input nodes, resulting in loss of input information. To avoid this degeneration, we also conduct a constrained random search that ensures all groups of input feature maps are used. This constraint is a type of human heuristics. After architecture derivation, we train these networks in the same way as previously described.

In Table II, we compare the test error of the derived architectures. The architecture produced by the full random search performs worst in both CIFAR-10/100. We note that one group (totally four groups) of input features in stage 3 is completely dropped in this architecture. Somewhat surprisingly, with the loss of input information, the results of the full random search are only slightly worse than those of the constrained random search. Overall, the architectures produced by PP-NAS perform better than random search, which proves the effectiveness of the proposed search algorithms.

However, we should also admit that the improvements of PP-NAS compared with randomly searched architecture in the constrained search space are minor through rigorous statistical analysis, which is also a common challenge for NAS [54]. We also notice that PP-NAS has obvious performance gain compared with randomly searched architectures in the full search space. As a result, we assume that with more runs of the experiments and fewer human priors for the search space design, PP-NAS may outperform random search obviously.

C. Searching on ImageNet

ImageNet-1K (ILSVRC2012) [55] dataset consists of 1.3 M images for training and 50 k images for testing, equally distributed among 1000 classes. Due to the use of one-level optimization, no extra validation set is split from the training set. Due to the low memory usage and training cost of PP-NAS, we directly search on the full training set without any subsampling.

The training protocol generally follows [56]. We use label smoothing ($\epsilon = 0.1$) as the regularization strategy and use SGD with a weight decay of 1e−4, a momentum of 0.9, and a mini-batch of 1024. We use a learning rate of 0.4 and adjust it according to a cosine schedule for training 120 epochs and with a warm-up of five epochs [57]. Mixup or knowledge distillation [58] is not used to avoid the long training time. The architecture parameters α and β are optimized by Adam, with a fixed learning rate of 1e−3 and a momentum (0.9, 0.999). Except for the use of Adam, all hyperparameters are the same between the search phase and the evaluation phase.

To further show the generalization of PP-ResNet, following the related works [59], [60], we also train it under the augmented setting. Specifically, the network is trained for 300 epochs with RandAugment [61] and Mixup as extra data augmentations. The RandAugment layer is set to 2, and the magnitude is set to 10. We also use stable weight decay [62] and reduce the value to 4e−5. Following [63], we train the network under the image size of 160 × 160 and evaluate under 224 × 224, but we do not fine-tune any layers after training. We also include other NAS methods to show the relative performance of PP-ResNet, part of which adopts a search space adapted from ResNet or DenseNet that has the similar derived architecture to PP-ResNet for a relatively fair comparison.

The architecture discovered on ImageNet is shown in Fig. 4. We compare the top-1 accuracy, top-5 accuracy, and the number of parameters of the discovered architecture with other models in Table III. Our PP-ResNet-50 has an improvement of 0.39% over Res2Net-50 on top-1 accuracy and 0.12% on top-5 accuracy with the same number of parameters. Under the augmented setting, PP-ResNet-50 can achieve an 80.58%

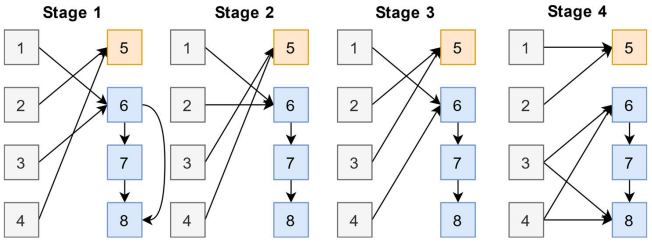

Fig. 4. Discovered blocks on ImageNet dataset for the four different stages of PP-ResNet-50.

TABLE III
IMAGE CLASSIFICATION RESULTS FOR IMAGENET DATASET

Architecture	Top-1 (%)	Top-5 (%)	Params (M)
DARTS (2nd) [24]	73.3	91.3	4.7
FairDARTS [35]	77.2	93.5	5.3
GOLD-NAS [30]	76.1	92.7	6.4
CyDAS [49]	76.3	92.9	6.1
DenseNAS [52]	78.0	-	24.7
GreedyNASv2-L [53]	81.1	95.4	26.9
ResNet-50	78.37	93.99	25.6
Res2Net-50 (26w×4s)	79.09	94.45	25.7
PP-ResNet-50 (26w×4s)	79.48	94.57	25.7
PP-ResNet-50† (26w×4s)	78.92	94.29	25.7
PP-ResNet-50* (26w×4s)	80.58	95.34	25.7

† *Directly derived after searching without retraining.*
* *Trained under the augmented setting.*

top-1 accuracy and 95.34% top-5 accuracy. It also outperforms comparing NAS methods, except for GreedyNASv2 [53], which uses evolutionary algorithms for searching. However, PP-ResNet is gradient-based and requires less time to search the architecture compared with GreedyNASv2, which somehow compensates for the performance limitation. Besides, the search space of PP-NAS focuses on the forward paths for convolution blocks, while GreedyNASv2 cares more about the combination of different existing convolution operations with different kernel sizes and widths. Since the focus is different, PP-NAS may be integrated into the search pipeline of other NAS methods in the future.

Interestingly, it is also possible to avoid retraining, considering the minor differences between the search and the evaluation in this work. At the end of search, all $\sigma(\alpha)$ are around 0.995 or 0.005, so we can safely prune these unnecessary connections. After pruning, PP-ResNet without retraining can achieve a top-1 accuracy of 78.92%, which is only 0.56% lower than retraining.

D. Object Detection

We further validate PP-ResNet on the large-scale detection benchmark COCO [64]. Following previous works [6], [7], [65], we use the COCO train2017 split (115 k images) for training and val2017 split (5 k images) as the main results.

TABLE IV
GFLV2-BASED OBJECT DETECTION RESULTS ON THE COCO DATASETS

Architecture	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-50	44.3	62.3	48.5	26.8	47.7	54.1
ResNet-50†	44.1	61.7	48.2	25.8	48.2	58.5
Res2Net-50	45.5	63.2	49.5	**27.4**	49.6	60.0
PP-ResNet-50	**46.1**	**64.1**	**50.1**	27.2	**50.4**	**61.4**

† *Our reimplemented version.*

GFLV2 [66] is one of the state-of-the-art one-stage detectors used as the baseline method with ResNet-50, Res2Net-50, and PP-ResNet-50 as the backbone network. For a fair comparison, we reimplement GFLV2 and keep all the implementation details the same except for the use of different backbone networks.

First, the EfficientDet [8] style data augmentation is used rather than the faster RCNN style [6]. Specifically, faster RCNN randomly resizes the short edge of the original image, while EfficientDet randomly resizes the original image and crops a square region from it. We use a crop size of 896 × 896, which is close to the original 800 × 1333. We use a resizing range from 0.5 to 2.0, following the implementation of EfficientDet. A random horizontal flip is also used after cropping. During testing, we resize and pad the images to the target size (896 × 896) without flipping or multiscale augmentation.

We train all networks with the SGD optimizer with a momentum of 0.9 and a weight decay of 1e−4. We use a total batch size of 32 on eight TPUv2 cores and a learning rate of 0.02. The learning rate is linearly warmed up from 0 to 0.02 for the first one epoch and then decayed to 0 according to a cosine schedule. Synchronized batch normalization is added after every convolution with a momentum of 0.9. All models are trained for 24 epochs (around 90 k iterations, comparable to 2× the schedule in [67]). GIoU [68] loss is used for the bounding box regression with a weight of 2.0. At inference, we keep the top 5 k predictions from all FPN levels [7] and then apply the standard nonmaximum suppression with an IoU threshold of 0.6 and a confidence threshold of 0.05 to yield the final detections.

Table IV shows the object detection results on COCO val2017. Note that our reimplemented GFLV2 with ResNet-50 backbone network has similar results as the original paper. Overall, the PP-ResNet-50-based model outperforms ResNet-50 and Res2Net-50 by 2.0% and 0.6% on average precision (AP). For specific metrics, PP-ResNet performs better on AP_{50}, AP_{75}, AP_M, and AP_L, and worse on AP_s, which indicates more accurate detection, especially for larger objects. It might be explained that larger objects benefit more from multiscale features.

PASCAL VOC [69] is another standard benchmark dataset for object detection, which contains 20 common object classes. We use VOC2007 trainval and VOC 2012 trainval

TABLE V
RETINANET-BASED OBJECT DETECTION RESULTS ON PASCAL VOC

Method	mAP	aero-plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor bike	per-son	plant	sheep	sofa	train	tv
ResNet	82.37	85.0	88.1	85.8	76.7	70.2	86.6	89.0	**90.1**	67.0	88.2	74.3	90.2	88.5	85.0	85.0	59.8	86.7	80.7	87.4	83.0
Res2Net	83.28	**88.0**	**88.8**	**87.6**	76.6	72.7	**88.5**	89.0	89.2	69.6	87.6	76.1	89.1	89.3	87.1	85.6	61.6	85.5	80.8	**88.3**	**85.0**
PP-ResNet	**83.82**	87.7	88.2	87.0	**77.1**	**74.7**	87.3	**89.2**	90.0	**70.3**	**88.7**	**79.3**	**90.2**	**89.3**	**87.4**	**86.3**	**62.0**	**88.2**	**80.9**	87.4	84.8

(16 551 images) for training and test on the VOC2007 test (4952 images), which follows the common practice [70], [71], [72].

RetinaNet [65] is used as the baseline method with ResNet-50, Res2Net-50, and PP-ResNet-50 as the backbone network. For a fair comparison, we keep all the implementation details the same and only use different backbone networks.

The data augmentation and training settings are almost the same as COCO with a few differences. For PASCAL VOC, we use a target size of 640 × 640. The L1 loss is used for bounding box regression with a weight of 1.0. For nonmaximum suppression, we use an IoU threshold of 0.5 to yield the final detections.

Table V shows the object detection results on the VOC2007 test set. Overall, the PP-ResNet-50-based model outperforms ResNet-50 and Res2Net-50 by 1.45% and 0.54%, respectively, on mean AP (mAP). For per class results, PP-ResNet-50 performs best on 13/20 of all classes, especially for those poorly detected classes (mAP < 80%). For example, PP-ResNet-50 outperforms Res2Net-50 by 2.0% for the bottle and 3.2% for the table.

E. Semantic Segmentation

Multiscale representations are essential for semantic segmentation, which is position-sensitive and relies on contextual information of objects. We thus evaluate our PP-ResNet on the semantic segmentation task using the PASCAL VOC dataset and the Cityscapes dataset [73].

Following previous works [74], [75], [76], we use the augmented PASCAL VOC 2012 dataset [77], which contains 10 582 images for training and 1449 images for validation.

We use DeepLabv3+ [76] as the segmentation method. We reimplement DeepLabv3+ and keep all the details the same except that the backbone network is replaced with ResNet, Res2Net, or our proposed PP-ResNet. The output strides used in training and evaluation are both 8. The multi-grid method of (1, 2, 4) is also used for better performance.

Following previous works [76], we employ a crop size to be 512 during both training and test on the PASCAL VOC 2012 dataset. For data augmentation, we randomly scale the input images (from 0.5 to 2.0), then randomly left-right flip the images, and finally randomly crop square patches from them during training. When testing, we only pad the original images to the target size (512 × 512) without resizing. Single-scale results are reported.

All models are trained with the SGD optimizer with a momentum of 0.9 and a weight decay of 1e−4. We use a total batch size of 16 on eight TPUv2 cores and a learning rate of 0.01. The learning rate is linearly warmed up from 0 to 0.01 for the first five epochs and then decayed to 0 according to a cosine schedule. Synchronized batch normalization is added after every convolution with momentum of 0.9. All models are trained for 60 epochs (around 40 k iterations).

TABLE VI
DEEPLABV3+ BASED SEMANTIC SEGMENTATION RESULTS ON PASCAL VOC AND CITYSCAPES

Dataset	Backbone	mIoU
PASCAL VOC	ResNet-50	78.74
	Res2Net-50	79.13
	PP-ResNet-50	**79.55**
Cityscapes	ResNet-50	77.99
	Res2Net-50	78.82
	PP-ResNet-50	**79.40**

Cityscapes is a large-scale dataset containing high-quality pixel-level annotations of 5000 images (2975, 500, and 1525 for training, validation, and test sets, respectively) and about 20 000 coarsely annotated images. We use these 2975 images for training and 500 images for validation.

The data augmentation is generally the same as VOC except for a different crop size of 512 × 1024 and additional random photometric distortion. When testing, we simply use the original images without flipping or multiscale augmentation.

The training settings are also almost the same as VOC except for a smaller batch size of 8 and longer training epochs of 90 (around 45 k iterations).

Table VI shows the semantic segmentation results on the PASCAL VOC dataset and the Cityscapes dataset. For PASCAL VOC, our PP-ResNet-50-based model outperforms ResNet-50 and Res2Net-50 by 0.81% and 0.42% on mean IoU (mIoU), respectively. For Cityscapes, the PP-ResNet-50-based model outperforms ResNet-50 and Res2Net-50 by 1.41% and 0.58%, respectively. The greater improvement in Cityscapes than VOC might be explained by images in the Cityscapes dataset being harder to segment and requiring stronger multiscale feature extraction ability, which is just the advantage of PP-ResNet.

Figs. 6 and 7 visualize the semantic segmentation results on the PASCAL VOC and Cityscapes dataset using PP-ResNet,

TABLE VII
LATENCY (ms/IMAGE) OF DIFFERENT RESNET-STYLE MODELS ON THREE CLASSIFICATION DATASETS

	C10†	C100†	ImageNet*
ResNet-110	0.42	0.42	-
ResNeXt-110 (4c×24w)	2.43	2.43	-
Res2Net-110 (26w×4s)	3.03	3.03	-
PP-ResNet-110 (26w×4s)	3.16	3.14	-
ResNet-50	-	-	0.34
Res2Net-50 (26w×4s)	-	-	1.09
PP-ResNet-50 (26w×4s)	-	-	1.10

† *Measured using a single Tesla T4 GPU, with a resolution of 32×32 and batchsize of 1024 on the average of 5 runs.*
* *Measured using a single Tesla T4 GPU, with a resolution of 224×224 and batchsize of 256 on the average of 5 runs.*

Res2Net, and ResNet as backbone networks. For the PASCAL VOC dataset, our PP-ResNet produces more complete segmentation results. For the Cityscapes dataset, PP-ResNet's segmentation results have finer details and less classification error.

V. DISCUSSION

A. Latency of PP-ResNets

Inference speed, or latency, is an important metric to measure the complexity and practicality of neural networks, especially on mobile devices that have limited calculation or under scenarios that require fast or real-time processing. However, NAS methods find out optimized architectures that usually have complex structures in the search spaces, which may severely increase the latency of the derived architecture.

To explore the factors that may affect the latency of PP-NAS, as well as showing the practicality of it for wide applications, we evaluate the latency of PP-ResNets on CIFAR-10/100 and ImageNet, respectively. In detail, latency is measured using a single Tesla T4 GPU with a batch size of 1024 for CIFAR and 256 for ImageNet. The batch size decrease is due to limited GPU memory. The results are reported on the average of five runs. We also evaluate the latency of ResNet, ResNeXt, and Res2Net under the same settings for a fair comparison. The results are shown in Table VII.

For experiments on CIFAR, due to the complex calculation of PPConv, the latency of PP-NAS increases compared with the original ResNet. However, it has a similar value compared with ResNeXt that has group convolutions and Res2Net that has both group features and inner connections for the output feature groups. The results indicated that the latency is mainly due to the groups and the cascade structures of the output features.

Similar results of latency are also reported on ImageNet, where the test resolution is set to 224 × 224. It should be noted that PP-ResNet slightly increases the inference cost compared with Res2Net, while the performance gain is obvious

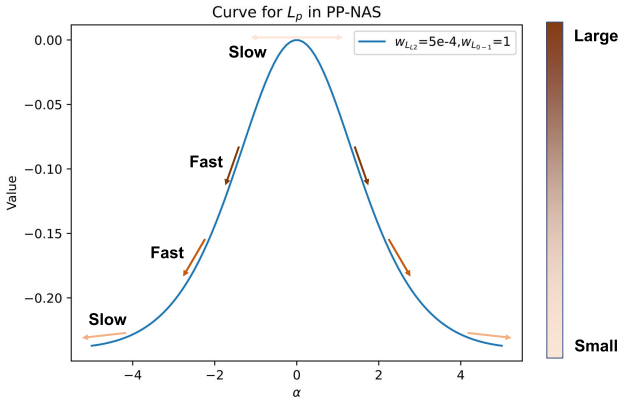

Fig. 5. Curve for $L_p = w_{L_{L2}} L_{L2} + w_{L_{0-1}} L_{0-1}$. Colored arrows show the direction of the gradient with respect to connection parameters α. The darker the color, the larger the absolute value of the gradient.

TABLE VIII
ABLATION STUDIES FOR LOSS TERMS OF PP-RESNET-110 EVALUATED ON CIFAR-10/100

	L_{CE}	L_{L2}	L_{0-1}	L_{conn}	Test Error (%)	
					C10	C100
PP-NAS	√	√	√	√	3.50	17.33
Ablations	√	√	√		3.56/Fail	17.69/Fail
	√	√		√	Fail	Fail
	√	√			Fail	Fail
	√				Fail	Fail

(0.39% Top-1 accuracy). These results show the effectiveness of PP-NAS in ResNet structures.

B. Ablation Studies for Loss Terms

Understanding the importance of the loss terms in (5) is important to better analyze PP-NAS. In this work, we introduce L_{0-1} and propose L_{conn} to promote the learning of the architecture parameters. We also use the commonly applied L_{L2} regularization to ensure that the learned values are suitable. To further show the necessity of these loss terms, we conduct ablation studies, which are evaluated on CIFAR-10/100. The weights for loss terms are set the same as experiments in Section IV-A. The derivation threshold $\sigma_{\text{threshold}}$ is also set to 0.9 for each ablation counterpart. We search three times and train for three times under each of these settings and provide the average results for successful architectures. The ablation results are shown in Table VIII.

For loss without L_{conn}, since in many of the cases, there exists at least one connection for each feature group within the PPConv, L_{conn} shows little effect in these cases. As a result, the architectures that are searched without it show competitive performance to the one we searched. However, we also find some degenerated architectures that have no connection to part of the output groups, which decreases the number of total output features and results in derivation failure.

For loss without L_{0-1}, we find that $\sigma(\alpha^{(i,j)})$ are almost of range (0.2, 0.8). Due to the large threshold for connection

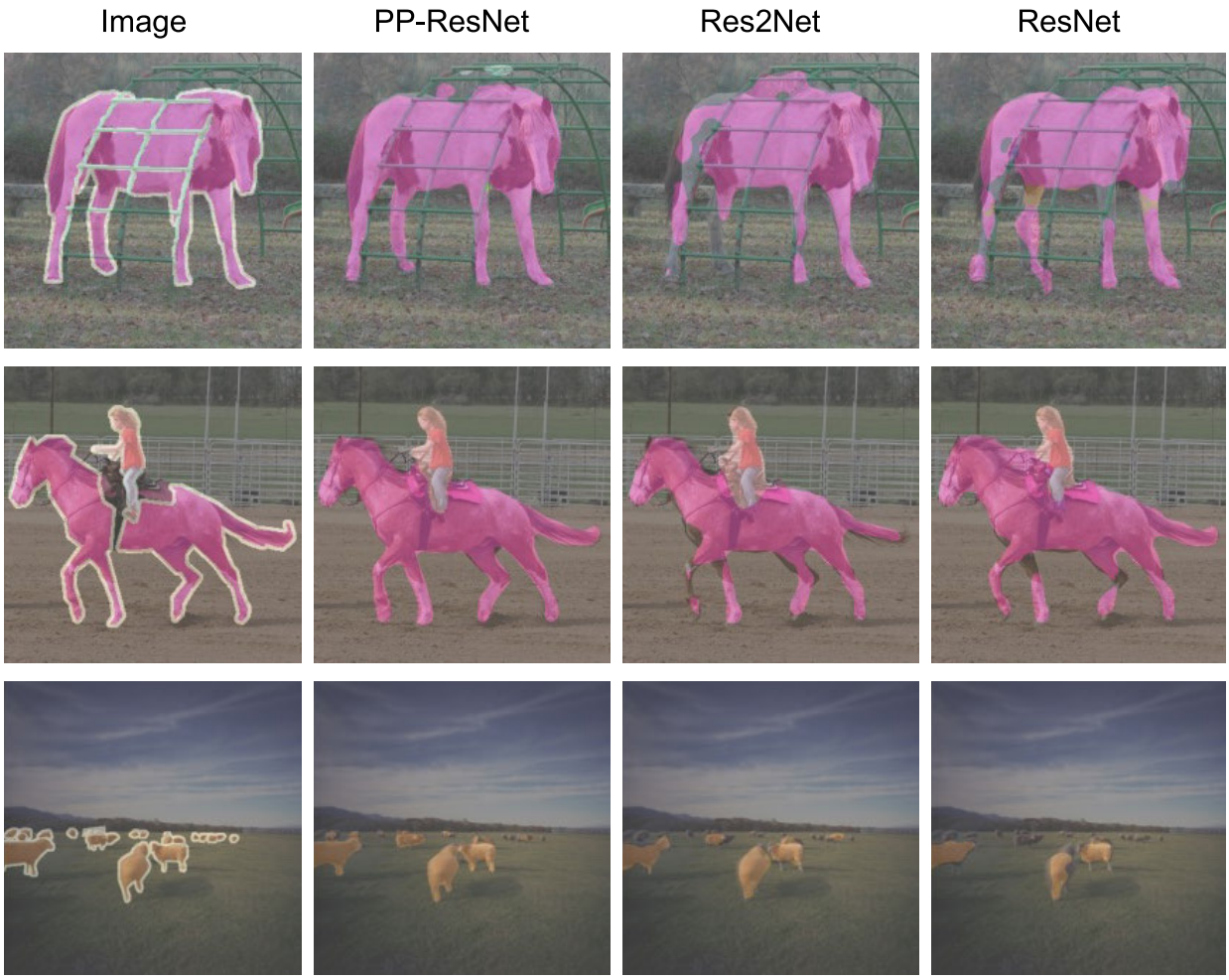

Fig. 6. Visualization of semantic segmentation results on the PASCAL VOC dataset using different backbone networks.

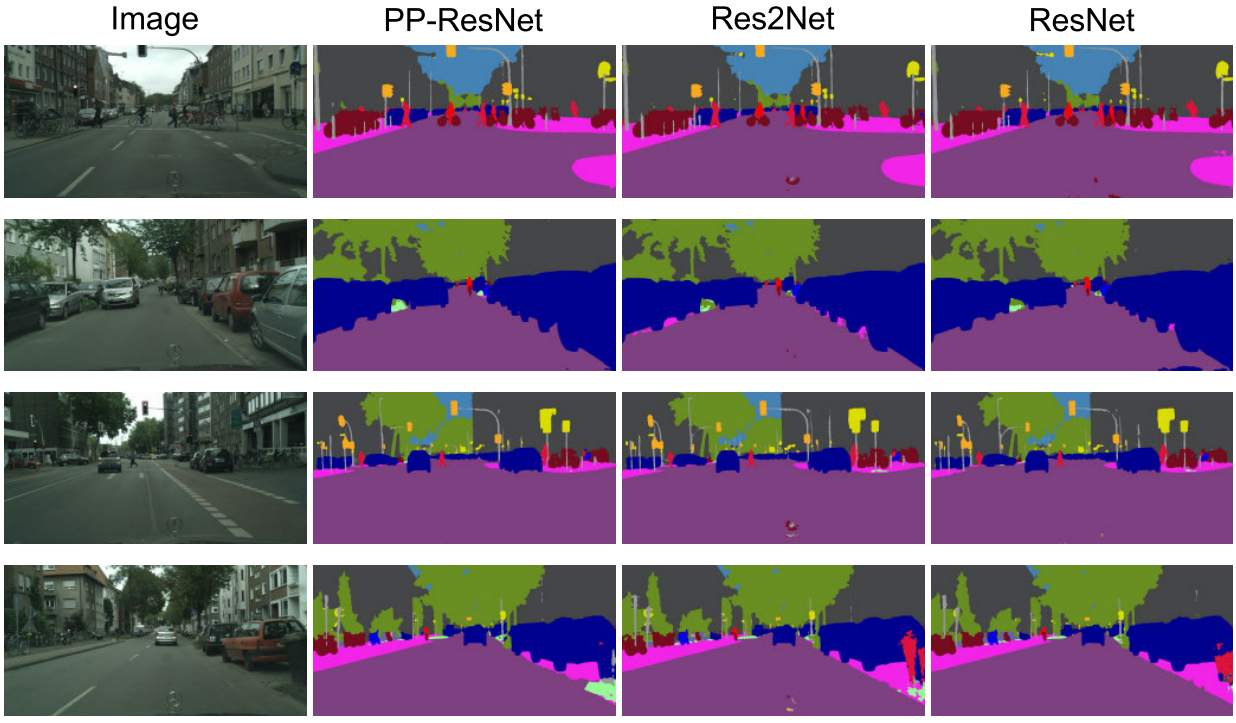

Fig. 7. Visualization of semantic segmentation results on the Cityscapes dataset using different backbone networks.

derivation, the experiments fail to derive architectures that can propagate the information. As a result, these experiments are failed. Decreasing $\sigma_{\text{threshold}}$ may work in these cases, while this violates the motivation for designing a method with a small optimization gap between super-net and searched architectures.

From the ablation experiments, we demonstrate that both L_{conn} and L_{0-1} are required to ensure the successful architecture, indicating that the design of these loss terms is necessary in the searching framework of PP-NAS.

C. How the Loss Affects Connection Parameter Learning

Connection parameters α determine the forward paths of PPConv, which are important and worth further analysis. To understand how PP-NAS learns the connection parameters α, we draw the partial loss $L_p = w_{L_{L2}} L_{L2} + w_{L_{0-1}} L_{0-1}$ in Fig. 5, where $w_{L_{L2}}$ and $w_{L_{0-1}}$ are the weights of L_{L2} and L_{0-1} with values of 5e−4 and 1, respectively. We omit L_{CE} due to the complex form of this loss term, which is related to the super-net architecture and is difficult to draw. Besides, as reported by FairDARTS [35], L_{CE} alone shows the limited effect on the learning of α. Therefore, the discussion of how loss affects learning α without L_{CE} is acceptable. We also ignore L_{conn} due to the motivation that it is designed to avoid degenerated architectures that have no connection to part of the feature groups. L_{conn} shows little effect in many of the cases. Note that the hyperparameter settings are the same as the ones in experiments on CIFAR.

As shown in Fig. 5, the learning of α can be roughly divided into three stages without a certain boundary, where at the beginning, the learning is slow due to the small gradient. However, this stage mainly determines the connections to keep. The middle stage has a large absolute value for the gradient, and thus, α changes rapidly. This stage shrinks the gap between the super-net and subarchitectures. It is also important, especially when we directly prune subarchitecture from the super-net without retraining the model weights. The learning of α slows down in the final stage, in which α combined with a proper value of $\sigma_{\text{threshold}}$ further refines the forward connections.

D. Ability for Multiscale Feature Learning

Our PP-NAS ensures the learning of multiscale features within one convolutional block, which is achieved by the design of our search space that includes cascade structures for operations within the block. With connections within output feature groups, the information is passed through different operations that may expand the receptive field. In detail, when the searched operation is 3×3 convolution or other operations with a kernel size larger than 1, the receptive field expands.

More operations with larger kernel sizes or different kernel shapes are worth trying to further promote the performance of PPConv. The design of the search space of PP-NAS is also compatible with more candidate operations. However, in our experiments, we simply choose the 3×3 convolution to expand the scale of features in order to be compatible with group convolutions and Res2Net convolutions for a relatively fair comparison.

E. Limitations

One of the limitations of PP-NAS is that the sigmoid activation function for connection parameters suffers from gradient killing that may cause an optimization gap when the activated value is close to 0 and 1. Thus, more activation functions and derivation strategy are worth trying in the future, which may further promote the performance of PP-NAS.

Another limitation of PP-NAS is that it learns architecture parameters based on the general features of the training dataset and treats all input data the same. As a consequence, some features that are important for part of the data, while unimportant for others, are computed on the same forward path, which may be suboptimal. Architecture customization may alleviate the problem, while customization is expensive for NAS due to large memory consumption and complex search space design, which is beyond the focus of this work. We will try architecture customization for data with different features in the future.

VI. Conclusion

This work proposed a novel PP-NAS method, which includes a new search space PPConv for plug-and-play blocks and the corresponding search algorithm. PPConv can be easily integrated into existing networks by replacing the main convolution operation. For simplicity, we mainly apply PP-NAS on ResNet architectures in this work and name it as PP-ResNet. PPConv search space decouples connections and operations, thus resulting in a lower memory usage and training cost. Our search algorithm uses one-level optimization to speed up and simplify the search procedure and introduces extra loss functions to help search. PP-NAS largely shrinks the optimization gap caused by weight sharing so that PP-ResNet with discovered novel blocks can outperform ResNet, ResNeXt, and Res2Net on numerous vision tasks, including image classification, object detection, and semantic segmentation.

In the future, we plan to integrate PPConv to more network architectures, especially lightweight networks such as MobileNet and ShuffleNet. In addition, with further optimization to PP-NAS, we believe that it will be possible to search on downstream tasks (detection and segmentation) directly and achieve better results. Finally, the proposed multiscale networks also have the potential to perform well in many other vision tasks, such as image super-resolution and denoising.

References

[1] S. H. Gao, M. M. Cheng, and K. Zhao, "Res2Net: A new multi-scale backbone architecture," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 2, pp. 652–662, Feb. 2021.

[2] M. Tan and Q. V. Le, "MixConv: Mixed depthwise convolutional kernels," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2019, pp. 1–13.

[3] I. C. Duta, L. Liu, F. Zhu, and L. Shao, "Pyramidal convolution: Rethinking convolutional neural networks for visual recognition," 2020, *arXiv:2006.11538*.

[4] P. Yuan et al., "HS-ResNet: Hierarchical-split block on convolutional neural network," 2020, *arXiv:2010.07621*.

[5] L. Zhou, Z. Wang, Y. Luo, and Z. Xiong, "Separability and compactness network for image recognition and superresolution," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 11, pp. 3275–3286, Nov. 2019.

[6] 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.

[7] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 936–944.

[8] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10778–10787.

[9] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019.

[10] K.-H. Shih, C.-T. Chiu, J.-A. Lin, and Y.-Y. Bu, "Real-time object detection with reduced region proposal network via multi-feature concatenation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 6, pp. 2164–2173, Jun. 2020.

[11] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 386–397, Feb. 2018.

[12] X. Chang, H. Pan, W. Sun, and H. Gao, "YolTrack: Multitask learning based real-time multiobject tracking and segmentation for autonomous vehicles," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5323–5333, Dec. 2021.

[13] 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. 2021.

[14] B. Cheng, B. Xiao, J. Wang, H. Shi, T. S. Huang, and L. Zhang, "HigherHRNet: Scale-aware representation learning for bottom-up human pose estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 5385–5394.

[15] L. Ke, M.-C. Chang, H. Qi, and S. Lyu, "Multi-scale structure-aware network for human pose estimation," 2018, *arXiv:1803.09894*.

[16] A. Borji, M.-M. Cheng, Q. Hou, H. Jiang, and J. Li, "Salient object detection: A survey," *Comput. Vis. Media*, vol. 5, pp. 117–150, Jun. 2019.

[17] Y. Ji, H. Zhang, Z. Jie, L. Ma, and Q. M. J. Wu, "CASNet: A cross-attention Siamese network for video salient object detection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 6, pp. 2676–2690, Jul. 2021.

[18] B. Shen et al., "Real-time intraoperative glioma diagnosis using fluorescence imaging and deep convolutional neural networks," *Eur. J. Nucl. Med. Mol. Imag.*, vol. 48, no. 11, pp. 3482–3492, Oct. 2021.

[19] Z. Hu et al., "First-in-human liver-tumour surgery guided by multispectral fluorescence imaging in the visible and near-infrared-I/II windows," *Nature Biomed. Eng.*, vol. 4, no. 3, pp. 259–271, Dec. 2019.

[20] B. Chang et al., "A phosphorescent probe for in vivo imaging in the second near-infrared window," *Nature Biomed. Eng.*, vol. 6, no. 5, pp. 629–639, Aug. 2021.

[21] A. Xiao et al., "Intraoperative glioma grading using neural architecture search and multi-modal imaging," *IEEE Trans. Med. Imag.*, vol. 41, no. 10, pp. 2570–2581, Oct. 2022.

[22] S. Kugunavar and C. J. Prabhakar, "Convolutional neural networks for the diagnosis and prognosis of the coronavirus disease pandemic," *Vis. Comput. Ind., Biomed., Art*, vol. 4, no. 1, p. 12, Dec. 2021.

[23] K. Jiang et al., "Multi-scale hybrid fusion network for single image deraining," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Sep. 24, 2021, doi: 10.1109/TNNLS.2021.3112235.

[24] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–4.

[25] Z. Ding, Y. Chen, N. Li, D. Zhao, Z. Sun, and C. L. P. Chen, "BNAS: Efficient neural architecture search using broad scalable architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 9, pp. 5004–5018, Sep. 2022.

[26] H. Tan et al., "RelativeNAS: Relative neural architecture search via slow-fast learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 1, pp. 475–489, Jan. 2023.

[27] Y. Zhou, X. Xie, and S.-Y. Kung, "Exploiting operation importance for differentiable neural architecture search," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 11, pp. 6235–6248, Nov. 2022.

[28] H. Wang, R. Yang, D. Huang, and Y. Wang, "IDARTS: Improving DARTS by node normalization and decorrelation discretization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 4, pp. 1945–1957, Apr. 2023.

[29] A. Zela et al., "Understanding and robustifying differentiable architecture search," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020, pp. 1–28.

[30] K. Bi, L. Xie, X. Chen, L. Wei, and Q. Tian, "GOLD-NAS: Gradual, one-level, differentiable," 2020, *arXiv:2007.03331*.

[31] B. Shen, A. Xiao, J. Tian, and Z. Hu, "PP-NAS: Searching for plug-and-play blocks on convolutional neural network," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2021, pp. 365–372.

[32] X. Chen and C.-J. Hsieh, "Stabilizing differentiable architecture search via perturbation-based regularization," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 1554–1565.

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

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

[35] X. Chu et al., "Fair DARTS: Eliminating unfair advantages in differentiable architecture search," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2020, pp. 465–480.

[36] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5987–5995.

[37] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 2261–2269.

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

[39] L. Xie et al., "Weight-sharing neural architecture search: A battle to shrink the optimization gap," *ACM Comput. Surv.*, vol. 54, no. 9, pp. 1–37, Dec. 2022.

[40] Y. Tian, C. Liu, L. Xie, J. Jiao, and Q. Ye, "Discretization-aware architecture search," *Pattern Recognit.*, vol. 120, Dec. 2021, Art. no. 108186.

[41] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[42] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2016, p. 1–15.

[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–22.

[44] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," 2017, *arXiv:1708.04552*.

[45] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le, "AutoAugment: Learning augmentation strategies from data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 113–123.

[46] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–16.

[47] H. Zhang et al., "mixup: Beyond empirical risk minimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–14.

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

[49] H. Yu et al., "Cyclic differentiable architecture search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 211–228, Jan. 2023.

[50] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.

[51] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," 2019, *arXiv:1902.07638*.

[52] J. Fang, Y. Sun, Q. Zhang, Y. Li, W. Liu, and X. Wang, "Densely connected search space for more flexible neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10628–10637.

[53] T. Huang et al., "GreedyNASv2: Greedier search with a greedy path filter," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 11902–11911.

[54] A. Yang, P. M. Esperança, and F. M. Carlucci, "NAS evaluation is frustratingly hard," 2019, *arXiv:1912.12522*.

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

[56] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of tricks for image classification with convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 558–567.
[57] P. Goyal et al., "Accurate, large minibatch SGD: Training ImageNet in 1 hour," 2017, *arXiv:1706.02677*.
[58] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2014, pp. 1–9.
[59] H. Zhang et al., "ResNeSt: Split-attention networks," 2021, *arXiv:2004.08955*.
[60] I. Bello et al., "Revisiting ResNets: Improved training and scaling strategies," 2021, *arXiv:2103.07579*.
[61] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2020, pp. 3008–3017.
[62] Z. Xie, I. Sato, and M. Sugiyama, "Understanding and scheduling weight decay," 2020, *arXiv:2011.11152*.
[63] H. Touvron et al., "Fixing the train-test resolution discrepancy," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2019, pp. 1–11.
[64] T.-Y. Lin et al., "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 740–755.
[65] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 2, pp. 318–327, Feb. 2020.
[66] X. Li, W. Wang, X. Hu, J. Li, J. Tang, and J. Yang, "Generalized focal loss v2: Learning reliable localization quality estimation for dense object detection," 2020, *arXiv:2011.12885*.
[67] K. Chen et al., "MMDetection: Open MMLab detection toolbox and benchmark," 2019, *arXiv:1906.07155*.
[68] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 658–666.
[69] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Sep. 2009.
[70] W. Liu et al., "SSD: Single shot MultiBox detector," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 21–37.
[71] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6517–6525.
[72] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li, "Single-shot refinement neural network for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4203–4212.
[73] M. Cordts et al., "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 3213–3223.
[74] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, Atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2017.
[75] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking Atrous convolution for semantic image segmentation," 2017, *arXiv:1706.05587*.
[76] L.-C. Chen et al., "Encoder–decoder with Atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 801–818.
[77] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 991–998.

Anqi Xiao is currently pursuing the Ph.D. degree with the CAS Key Laboratory of Molecular Imaging, Beijing Key Laboratory of Molecular Imaging, Institute of Automation, Chinese Academy of Sciences, Beijing, China, supervised via Prof. Zhenhua Hu.

Her research interests include deep learning and its application in medical images.

Biluo Shen received the master's degree from the Beijing Key Laboratory of Molecular Imaging, Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2022.

His research interests include deep learning, computer vision, and neural architecture search.

Jie Tian (Fellow, IEEE) received the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 1993.

He is the Chief Scientist with the CAS Key Laboratory of Molecular Imaging, Beijing Key Laboratory of Molecular Imaging, Institute of Automation, Chinese Academy of Sciences, and the Beijing Advanced Innovation Center for Big Data-Based Precision Medicine, School of Engineering Medicine, Beihang University, Beijing. His research interests are in optical multimodality molecular imaging technology development and artificial intelligence (AI) in radiomics.

Zhenhua Hu (Senior Member, IEEE) received the Ph.D. degree from Xidian University, Xi'an, Shaanxi, China, in 2012.

She is a Professor at the CAS Key Laboratory of Molecular Imaging, Beijing Key Laboratory of Molecular Imaging, Institute of Automation, Chinese Academy of Sciences, Beijing, China. Her research interests are in Cerenkov luminescence imaging and tomography, and fluorescence-guided surgeries.