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

Biluo Shen[1,2], Anqi Xiao[1,2], Jie Tian*[1,2,3] and Zhenhua Hu*[1,2]

[1]Beijing Key Laboratory of Molecular Imaging, Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
[3]Beijing Advanced Innovation Center for Big Data-Based Precision Medicine, Beihang University
{shenbiluo2019,xiaoanqi2020,zhenhua.hu,jie.tian}@ia.ac.cn

Abstract

Multi-scale features are of great importance in modern convolutional neural networks and show consistent performance gains on many vision tasks. Therefore, many plug-and-play blocks are introduced to upgrade existing convolutional neural networks for stronger multi-scale representation ability. However, the design of plug-and-play blocks is getting 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. Specifically, we design a new search space and develop the corresponding search algorithm. Extensive experiments on CIFAR10, CIFAR100, and ImageNet show that PP-NAS can find a series of novel blocks that outperform manually designed ones. Transfer learning results on representative computer vision tasks including object detection and semantic segmentation further verify the superiority of the PP-NAS over the state-of-the-art CNNs (e.g., ResNet, Res2Net). Our code will be made avaliable at https://github.com/sbl1996/PP-NAS.

1. Introduction

Multi-scale 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 maybe all helpful for understanding the object. Furthermore, under some circumstances, recognition of an object may be hard from the object itself, but much easier when relying on essential context information from multi-scale. Thus, it is of great importance to capture multi-scale features benefiting computer vision tasks including image classification [12, 26, 11, 30], object detection [24, 19, 27], instance segmentation [15], semantic segmentation [28, 6], keypoint detection [17], and salient object detection [2].

*Corresponding Authors

The key to capturing multi-scale features in convolutional neural networks is the design of network architecture. Many plug-and-play blocks [12, 26, 11, 30], which can be easily integrated into existing networks by replacing the regular convolution operation, were proposed to improve the multi-scale representation ability of 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 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 convolutional operation. Therefore, we name the new search space as 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 3x3 convolution with the searchable plug-and-play blocks. The concrete connections and operations inside the plug-and-play blocks will be searched and derived.

After the definition of search space, differentiable neural architecture search (DNAS, or DARTS) [22] methods are used to jointly optimize the network weights and architectural parameters in a weight-sharing super-net via gradient ascent. The final architecture is derived from the trained super-net at the end of the search phase. Different from most DARTS-based methods using bi-level optimization which suffers from heavy computational burden and inaccurate estimation of architectural gradients, we apply one-level optimization to speed up and simplify the search procedure and add strong regularization to prevent the failure causing by overfitting [32, 1].

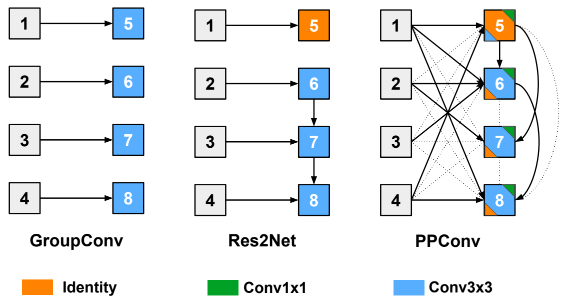

Figure 1. Comparison between group convolution (GroupConv), the blocks used by Res2Net, and our proposed searchable PPConv. The solid lines represent the selected connections, and the dotted lines represent the pruned connections. The selected operation is placed at the center of the node and the pruned operations are at the corner. The candidate operations include identity, 1x1 convolution and 3x3 convolution.

The overall NAS method containing both a novel search space and a corresponding search algorithm is named PP-NAS. We mainly apply PP-NAS on ResNet architectures and name it as PP-ResNet. We perform experiments on many representative vision tasks including image classification, object detection and semantic segmentation. PP-NAS can find a series of novel blocks that are similar but different from previous manually designed ones. PP-ResNet with the discovered blocks outperforms state-of-the-art CNNs (e.g., ResNet, ResNeXt, Res2Net), and shows consistent gains on datasets and benchmarks including CIFAR10/100, ImageNet, COCO, PASCAL VOC and Cityscapes.

2. Related Work

2.1. Multi-scale network and plug-and-play block

The key to capturing multi-scale features in convolutional neural networks is the design of network architecture. [12] proposed to connect split small filter groups in a hierarchical residual-like style to increase multi-scale representation strength. [26] mixed up multiple kernel sizes in a single depthwise convolution to capture patterns at different resolutions for better accuracy and efficiency. [11] introduced pyramidal convolution which contains a pyramid of kernels with varying size and depth to capture different levels of details in the scene. [30] addressed the Hierarchical-Split Block which contains many hierarchical splits and concatenates connections within a single residual block.

2.2. Differentiable Neural Architecture Search

Differentiable architecture search is one of the one-shot search methods, in which an over-parameterized super-network containing all candidate operations was trained only once. [22] introduced a differentiable framework by relaxing the search space so that the architectural parameters can be continuous and optimized together with the network weights by gradient descent. Despite its simplicity, many follow-on works reveal some of its drawbacks, such as instability, the inevitable aggregation of skip connections and the gap between the search and the evaluation. [5] designed a progressive search strategy to bridge the depth gap between the super-network and the sub-network. [7] proposed a zero-one loss combined with sigmoid function to alleviate the issue of discretization discrepancy. [32] showed that adding regularization strategies can robustify DARTS to find solutions with less curvature and better generalization performance. [1] 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.

3. Methodology

3.1. Search Space

The structure of the PPConv is shown in Figure 1. To make it easier to understand, we also show group convolutions [29] and the blocks used by Res2Net. Group convolutions split the feature maps into D groups, apply convolution on each group and concatenate all output feature maps. The Res2Net blocks add connections between output feature maps 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 groups.

PPConv is a generalization of group convolution and Res2Net blocks. Let's first split the input feature maps into s groups evenly and denote them as $x^{(i)}(0 < i \leq D)$, then we also have s groups of intermediate feature maps denoted as $x^{(i)}(D < i \leq 2D)$, and s groups of output feature maps denoted as $x'^{(i)}(D < i \leq 2D)$. A group of feature maps will be called a node for convenience in the following. The connection between x^i and x^j is a directed edge (i, j). Each intermediate node is computed based on all of its predecessors: $x^{(j)} = \sum_{i<j} x^{(i)}, D < j \leq 2D$, and each output node is the output of the associated operation applied on the intermediate node: $x'^{(j)} = o^{(j)}(x^{(j)}), D < j \leq 2D$. With this design of search space, the task of 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 is different from previous works.

The candidate operations include identity, 1x1 convolution and 3x3 convolution. By including these operations, we ensure that group convolution and Res2Net blocks are also in the search space. Zero operation is not included as candidates, because that we want to fix the number of output nodes and the number of output feature maps the same as the input.

Next, we describe the macro architecture. PPConv is designed to replace the main convolution (usually 3x3 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 3x3 convolution with stride 1 in the bottleneck is replaced with PPConv, for all the following experiments. We call the ResNet with PPConv as PP-ResNet.

To achieve better performance on different datasets, ResNet architectures with an appropriate number of stages and times of downsampling are required. Therefore, the concrete architecture is dependent on the dataset following the original ResNet. For example, the ResNets on CIFAR have 3 stages and downsample 2 times, and the ResNets on ImageNet have 4 stages and downsample 5 times. The PPConv structures are searched separately across different stages and shared between blocks in the stage.

3.2. 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 node $x^{(i)}$ 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, the sigmoid activation (σ) is used and the edges of a node will cooperate with each other, leading to a smoother information flow. More importantly, each edge will be switched on or off independently according to its corresponding parameter. Formally, the intermediate node is computed as:

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

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 discuss how to search operations. Let $\mathcal{O}$ denote a set of possible operations (identity, 1x1 convolution and 3x3 convolution) where every operation is to be applied on the intermidiate nodes to get output nodes. The categorical choice of a particular operation for $x'^{(j)}$ is relaxed to a softmax over all candidates:

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

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

After relaxation, we jointly optimize the architecture α, β and the network weights w. Previous differentiable methods recognized it as a bi-level 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 by, bi-level optimization suffers from heavy computational burden and inaccurate estimation of architectural gradients. We instead use one-level optimization that optimizes the architecture α, β and the weights w at the same time, 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 α, β and the weights correspondingly.

3.3. Architecture Derivation and Optimization Gap

At the end of the search when the architecture α, β is 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 $\sigma_{threshold}$. As for operations, we simply choose the one with the largest weight β.

As is well-known, the biggest pitfall of weight-sharing methods is the optimization gap between the super-net and the sub-architectures. Without any 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 is explicitly added to push the sigmoid value of architectural parameters towards 0 or 1, formally as,

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

After this, it is also possible that no connection is selected from the input nodes or to the intermidiate nodes if their associated $\sigma(\alpha)$ are all below $\sigma_{threshold}$. 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, formally as,

$$L_{conn} = \sum_i \max(1 - \sum_j \sigma(\alpha^{(i,j)}), 0)^2 + \sum_j \max(1 - \sum_{i<j} \sigma(\alpha^{(i,j)}), 0)^2 \quad (4)$$

As the zero-one loss pushes $\sigma(\alpha)$ towards 0 or 1, the connection loss should work well.

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

$$L = L_{CE} + L_{L2} + L_{0-1} + L_{conn} \quad (5)$$

Weights for the 4 parts of losses may be assigned and tuned, but we omit them here for clarity.

Our goal is to bridge the optimization gap and derive discrete architectures from the super-net without discretization error. The design of loss functions helps to eliminate it, but the depth and width gap still exists. We noticed that PPConv decouples the search of connections and operations, leading to a much less usage of memory and searching cost. Then it is possible to use precisely the same depth and width for architecture search as architecture evaluation. As a result, we can directly obtain a trained network by pruning unimportant connections and operations at the end of search, with a very small performance drop.

4. Experiments and Results

4.1. Searching on CIFAR

CIFAR-10 is a standard image classification dataset and consists of 50K training images with 5K images per class and 10K testing images with 1K images per class. The resolution of each image is 32×32. CIFAR-100 is just like CIFAR-10 and has the same images as CIFAR-10 but with 100 fine-grained classes. The training set of CIFAR-100 has 50K images with 500 images per class, and the test set has 10K 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 conducted 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. It should be noticed that, to achieve comparable performance with WRN [31], some improvements are applied. In the original paper, ResNet-110 is based on the basic blocks which are different from ours, so every stage in our ResNet-110 has 12 residual blocks. And the number of channels of 3 stages of ResNet-110 in the original paper is 16, 32, 64, but we use 64, 128, 256. The resulting ResNet-110 has a total of 18.5M parameters. Another tweak to the architecture is adding a 2x2 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, Res2Net for a fair comparison.

Architecture	Test Error (%)		Params (M)
	C10	C100	
SENet + Shake-Shake[1]	2.12	13.81	26.2
PyramidNet272 + Shakedrop[1]	1.70	11.70	26.0
ResNet-110	3.77	18.13	18.1
ResNeXt-110 (4×24d)	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

Table 1. Results of different architectures on CIFAR-10/100. The results denoted with 1 are trained for 1800 epochs. The result denoted with * is trained under the augmented setting (600 epochs).

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 0.1, momentum 0.9, and weight decay 5e-4, and the architecture parameters α are optimized by Adam [18], with a learning rate 1e-3, momentum (0.9, 0.999), with 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 an additional data augmentation including Cutout [10] and AutoAugment [9]. The learning rate for SGD is adjusted according to a cosine schedule [23] and the learning rate for Adam is fixed.

In the evaluation, the network depth and width are not changed, and the architecture is derived from the super-net. For CIFAR10/100, we have 2 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 augment setting, the network is trained for 600 epochs with Cutout, AutoAugment and Mixup [33]. The cutout length is 16 and the mixup ratio α is 0.2. Other hyperparameters are precisely the same as in the search phase. For each type of network architecture, we repeat the evaluation 5 times under the standard settings and 3 times under the augmented setting. The mean of these results is reported.

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

To ensure that PP-ResNet has similar number of parameters with ResNeXt and Res2Net, we keep exactly 1 identity operation after searching. This decision slightly narrows search space and may affect final performance. Similarly, many previous works of DARTS[5, 21] limited the number of "skip connect" operation to 1 or 2.

In Table 1, we compare the test error and the number of parameters of the discovered architectures with other models. Our PP-ResNet has an improvement of 0.17% over Res2Net on CIFAR-10 and 0.13% on CIFAR-100 under the standard setting.

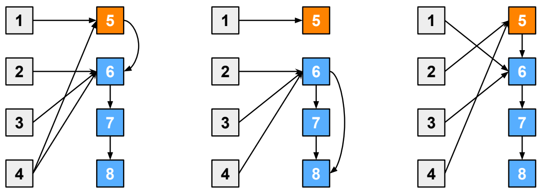

Figure 2. The discovered block on CIFAR-10 for the 3 different stages of PP-ResNet.

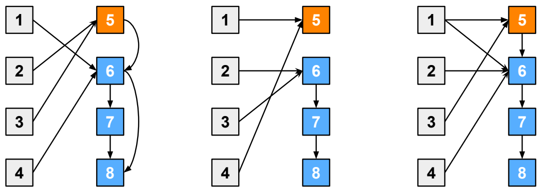

Figure 3. The discovered block on CIFAR-100 for the 3 different stages of PP-ResNet.

The architectures discovered on CIFAR-10/100 are shown in Figure 2 and Figure 3. We find that there is no Conv1x1 in the discovered architectures. According to many related works of NAS, we speculate that the non-parametric identity operation has some special advantages over the parametric convolution operations during optimization. And between the parametric operations, Conv3x3 has 8x more parameters than Conv1x1, resulting in a big advantage. We plan to explain it from more aspects in the future.

4.2. Searching on ImageNet

ImageNet-1K(ILSVRC2012) dataset consists of 1.3M images for training and 50K images for testing, equally distributed among 1,000 classes. Thanks to the use of one-level optimization, no extra validation set is split from the training set. And because of 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 [16]. We use label smoothing ($\epsilon = 0.1$) as the regularization strategy, and use SGD with weight decay 1e-4, momentum 0.9, and a mini-batch of 1024. Our learning rates are adjusted according to a cosine schedule for training 120 epochs and with a warmup up of 5 epochs [13]. Mixup or Knowledge Distillation is not used to avoid the long training time. The architecture parameters α are optimized by Adam, with a fixed learning rate 1e-3 and momentum (0.9, 0.999). Except for the use of Adam, all hyperparameters are the same between the search phrase and the evaluation phase.

Architecture	Top-1	Top-5	Params
ResNet-50	78.37	93.99	25.6
Res2Net-50 (26wx4s)	79.09	94.45	25.7
PP-ResNet-50 (26wx4s)	79.48	94.57	25.7
PP-ResNet-50† (26wx4s)	78.92	94.29	25.7
PP-ResNet-50* (26wx4s)	**80.06**	**94.96**	25.7

Table 2. Results of different architectures on ImageNet. The result denoted with † is directly derived after searching without retraining. The result denoted with * is trained under the augmented setting (200 epochs).

The architecture discovered on ImageNet is shown in Figure 4. We compare the top-1 accuracy and top-5 accuracy and the number of parameters of the discovered architecture with other models in Table 2. Our PP-ResNet-50 has an improvement of 0.39% over Res2Net-50 on top-1 accuracy and 0.14% on top-5 accuracy with the same number of parameters. If trained for 200 epochs with AutoAugment and Mixup as extra data augmentations, PP-ResNet-50 can achieve an 80.06% top-1 accuracy and 94.96% top-5 accuracy.

Interestingly, it is also possible to avoid retraining, considering the minor differences between the search and the evaluation. 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.

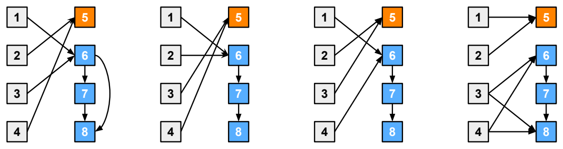

Figure 4. The discovered block on ImageNet for the 4 stages of PP-ResNet-50.

4.3. Object Detection

We further validate on the large-scale detection benchmark COCO. Following the previous works [24, 19], we use the COCO train2017 split (115K images) for training and val2017 split (5K images) as the major results.

GFLV2 [20] is one of the state-of-the-art one-stage detectors and used as the baseline method with ResNet-50, Res2Net-50 and PP-ResNet-50 as the backbone network. For a fair comparison, we reimplemented GFLV2 and keep all the implementation details the same except for the use of different backbone networks.

First, we use the EfficientDet style data augmentation rather than the Faster RCNN style. Specifically, Faster RCNN randomly resizes the short edge of the original image, while EfficientDet randomly resizes the original image and crop 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 (896x896) without flipping or multi-scale augmentation.

We train all networks with the SGD optimizer with momentum 0.9 and weight decay 1e-4. We use a total batch size of 32 on 8 TPUv2 cores and a learning rate of 0.02. The learning rate is linearly warmed up from 0 to 0.02 for the first 1 epoch and then decayed to 0 according to a cosine schedule. Synchronized batch normalization is added after every convolution with momentum 0.9. All models are trained for 24 epochs (around 90K iterations, comparable to 2x schedule). The GIoU [25] loss is used for bounding box regression with a weight of 2.0. At inference, we keep the top 5k predictions from all FPN levels and then apply the standard non-maximum suppression with an IoU threshold of 0.6 and confidence threshold 0.05 to yield the final detections.

Table 3 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 AP50, AP75, APM, and APL, and worse on APs, which indicates more accurate detection, especially for larger objects. It might be explained that larger objects benefit more from multi-scale features.

4.4. Semantic Segmentation

Multi-scale 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 PASCAL VOC dataset and Cityscapes dataset [8].

Following previous works [3, 4], we use the augmented PASCAL VOC 2012 dataset [14] which contains 10582 images for training and 1449 images for validation.

We use the DeepLabv3+ as the segmentation method.

Backbone	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
Re2sNet-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**

Table 3. GFLV2-based object detection results on the COCO datasets, measured using AP (%), AP@IoU=0.5 (%), AP@IoU=0.75 (%) and AP with different sizes. The result denoted with † is our reimplemented version, to ensure a fair comparison between different results.

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**

Table 4. DeepLabv3+ based semantic segmentation results on the PASCAL VOC dataset and Cityscapes dataset, measured using mIoU (%).

We reimplemented DeepLabv3+ and keep all 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, we employ crop size to be 512 during both training and test on 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 (512x512) without resizing. Single-scale results are reported.

All models are trained with the SGD optimizer with momentum 0.9 and weight decay 1e-4. We use a total batch size of 16 on 8 TPUv2 cores and a learning rate of 0.01. The learning rate is linearly warmed up from 0 to 0.01 for the first 5 epochs and then decayed to 0 according to a cosine schedule. Synchronized batch normalization is added after every convolution with momentum 0.9. All models are trained for 60 epochs (around 40K iterations).

Cityscapes is a large-scale dataset containing high-quality pixel-level annotations of 5000 images (2975, 500, and 1525 for the training, validation, and test sets respectively) and about 20000 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 512x1024 and additional random photometric distortion. When testing, we simply use the original images without flipping or multi-scale 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 45k iterations).

Table 5 shows the semantic segmentation results on PASCAL VOC dataset and Cityscapes dataset. For PASCAL VOC, our PP-ResNet-50 based model outperforms ResNet-50 and Res2Net-50 by 0.81% and 0.6% on mean IoU (mIoU). And for Cityscapes, the PP-ResNet-50 based model outperforms ResNet-50 and Res2Net-50 by 1.41% and 0.58%. The greater improvement in Cityscapes than VOC might be explained that images in Cityscapes dataset are harder to segment and require stronger multi-scale feature extraction ability, which is just the advantage of PP-ResNet.

5. 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. We applied PP-NAS on ResNet architectures to replace the main 3x3 convolution and obtained 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 many vision tasks including image classification, object detection, and semantic segmentation.

6. Acknowledgments

This study was supported by the National Key Research and Development Program of China (2017YFA0205200), National Natural Science Foundation of China (NSFC) (62027901, 92059207, 81930053, 81227901), Beijing Natural Science Foundation (JQ19027), the innovative research team of high-level local universities in Shanghai and the Zhuhai High-level Health Personnel Team Project (Zhuhai HLHPTP201703).

References

[1] Kaifeng Bi, Lingxi Xie, Xin Chen, Longhui Wei, and Qi Tian. GOLD-NAS: Gradual, One-Level, Differentiable. *arXiv*, pages 1–14, 2020. 2

[2] Ali Borji, Ming Ming Cheng, Qibin Hou, Huaizu Jiang, and Jia Li. Salient object detection: A survey. *Computational Visual Media*, 5(2):117–150, 2019. 1

[3] Liang-Chieh Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40:834–848, 2018. 6

[4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 6

[5] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-Octob, pages 1294–1303, 2019. 2, 5

[6] Bowen Cheng, Bin Xiao, Jingdong Wang, Honghui Shi, Thomas S. Huang, and Lei Zhang. HigherhrNet: Scale-aware representation learning for bottom-up human pose estimation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 5385–5394, 2020. 1

[7] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair DARTS: Eliminating Unfair Advantages in Differentiable Architecture Search. 2019. 2

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 6

[9] Ekin D. Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 113–123. Computer Vision Foundation / IEEE, 2019. 4

[10] Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017. 4

[11] Ionut Cosmin Duta, Li Liu, Fan Zhu, and Ling Shao. Pyramidal Convolution: Rethinking Convolutional Neural Networks for Visual Recognition. pages 1–16, 2020. 1, 2

[12] S. H. Gao, M. M. Cheng, K. Zhao, X. Y. Zhang, M. H. Yang, and P. Torr. Res2net: A new multi-scale backbone architecture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(2):652–662, 2021. 1, 2

[13] Priya Goyal, Piotr Dollár, Ross B. Girshick, P. Noordhuis, L. Wesolowski, Aapo Kyrola, Andrew Tulloch, Y. Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *ArXiv*, abs/1706.02677, 2017. 5

[14] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *2011 International Conference on Computer Vision*, pages 991–998, 2011. 6

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):386–397, 2020. 1

[16] Tong He, Zhi Zhang, H. Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 558–567, 2019. 5

[17] Lipeng Ke, Ming Ching Chang, Honggang Qi, and Siwei Lyu. Multi-Scale Structure-Aware Network for Human Pose Estimation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11206 LNCS:731–746, 2018. 1

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 4

[19] Xiaohan Li, Taotao Lai, Shuaiyu Wang, Quan Chen, Changcai Yang, and Riqing Chen. Weighted feature pyramid networks for object detection. In *Proceedings - 2019 IEEE Intl Conf on Parallel and Distributed Processing with Applications, Big Data and Cloud Computing, Sustainable Computing and Communications, Social Computing and Networking, ISPA/BDCloud/SustainCom/SocialCom 2019*, pages 1500–1504, 2019. 1, 5

[20] Xiang Li, Wenhai Wang, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang. Generalized focal loss v2: Learning reliable localization quality estimation for dense object detection. *arXiv preprint arXiv:2011.12885*, 2020. 6

[21] H. Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. Darts+: Improved differentiable architecture search with early stopping. *ArXiv*, abs/1909.06035, 2019. 5

[22] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *7th International Conference on Learning Representations, ICLR 2019*, pages 1–13, 2019. 1, 2

[23] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017*, 2017. 4

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017. 1, 5

[25] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union. June 2019. 6

[26] Mingxing Tan and Quoc V. Le. MixConv: Mixed depthwise convolutional kernels. In *30th British Machine Vision Conference 2019, BMVC 2019*, 2020. 1, 2

[27] Mingxing Tan, Ruoming Pang, and Quoc V Le. EfficientDet: Scalable and Efficient Object Detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020. 1

[28] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition, 2019. 1

[29] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 5987–5995, 2017. 2

[30] Pengcheng Yuan, Shufei Lin, Cheng Cui, Yuning Du, Ruoyu Guo, Dongliang He, Errui Ding, and Shumin Han. HS-ResNet: Hierarchical-split block on convolutional neural network, 2020. 1, 2

[31] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 4

[32] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and Robustifying Differentiable Architecture Search. In *8th International Conference on Learning Representations, ICLR 2020*, sep 2020. 2

[33] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. 4