

Improving Ranking Correlation of Supernet with Candidates Enhancement and Progressive Training

Ziwei Yang¹, Ruyi Zhang¹, Zhi Yang¹, Xubo Yang¹, Lei Wang³ and Zheyang Li^{1,2}

¹Hikvision Research Institute, ²Zhejiang University

³University of Science and Technology of China

{yangziwei5, zhangrui5, yangzhi13, yangxubo, lizheyang}@hikvision.com

wangl26@mail.ustc.edu.cn

Abstract

One-shot neural architecture search (NAS) applies weight-sharing supernet to reduce the unaffordable computation overhead of automated architecture designing. However, the weight-sharing technique worsens the ranking consistency of performance due to the interferences between different candidate networks. To address this issue, we propose a candidates enhancement method and progressive training pipeline to improve the ranking correlation of supernet. Specifically, we carefully redesign the sub-networks in the supernet and map the original supernet to a new one of high capacity. In addition, we gradually add narrow branches of supernet to reduce the degree of weight sharing which effectively alleviates the mutual interference between sub-networks. Finally, our method ranks the 1st place in the Supernet Track of CVPR2021 1st Lightweight NAS Challenge. Our code is released on https://github.com/Tend93/CVPR2021_NAS_competition_Track1_1st_solution.

1. Introduction

Neural architecture search aims to automatically design neural architectures. Although the architectures found by NAS methods [1, 3, 8] outperform human designed ones in many computer vision tasks, the previous methods [2, 7] require evaluating an enormous amount of candidate architectures with stand-alone training. The unaffordable computation overhead of the evaluation leads to a weight-sharing strategy of NAS[3, 5].

As one of the widely-used weight-sharing methods, one-shot NAS utilizes a supernet subsuming all candidate architectures within the search space to evaluate accuracies on the validation dataset. Instead of stand-alone training, all architectures directly inherit their weights from the supernet which is only trained once. The computation cost of

evaluation for architectures is effectively reduced.

To train supernet, NAS methods [4, 5] sample one or a few sub-networks from supernet and train them in each update step. Due to the weight-sharing fashion, sub-networks interfere with each other and exhibit inferior accuracies compared with stand-alone training. The accuracy ranking of sub-networks evaluated by the weights inheriting is inconsistency with the accuracy ranking of sub-networks when they are trained from scratch independently[1, 3].

For this issue, recent NAS methods attempt to improve the ranking correlation of supernet from two perspectives: optimizing the training process of supernet and enhancing the capacity of supernet by sub-network redesigning. FairNAS [4] employs a fair sampling strategy of sub-networks to increase the training accuracy. OFA [1] narrows the accuracy gap of sub-networks between supernet evaluating and stand-alone training by knowledge distilling. Although these methods can improve the ranking correlation of supernet in regular search spaces, it is difficult for training an irregular search space which contains searchable options with extreme conflicts, such as channel options 4 and 64. Furthermore, SCARLET-NAS[3] adds extra weights to improve the capacity of supernet for better convergence and less interferences between sub-networks. Laube et al.[6] trains the sub-networks independently by adding bias weights and splitting path weights. These methods alleviate the disturbances of sub-networks, but the additional weights need carefully designs and hyper-parameters tuning.

In this paper, we apply candidates enhancing and progressive training pipeline to improve the ranking consistency of candidate architectures evaluated via supernet and stand-alone training. We find some sub-networks in the supernet exhibit inferior accuracies in the joint optimization of all sub-networks. It contributes to the degradation of ranking correlation between supernet evaluating and stand-alone training. To tackle this problem, we convert the sub-networks to the ones of higher capacity, which only

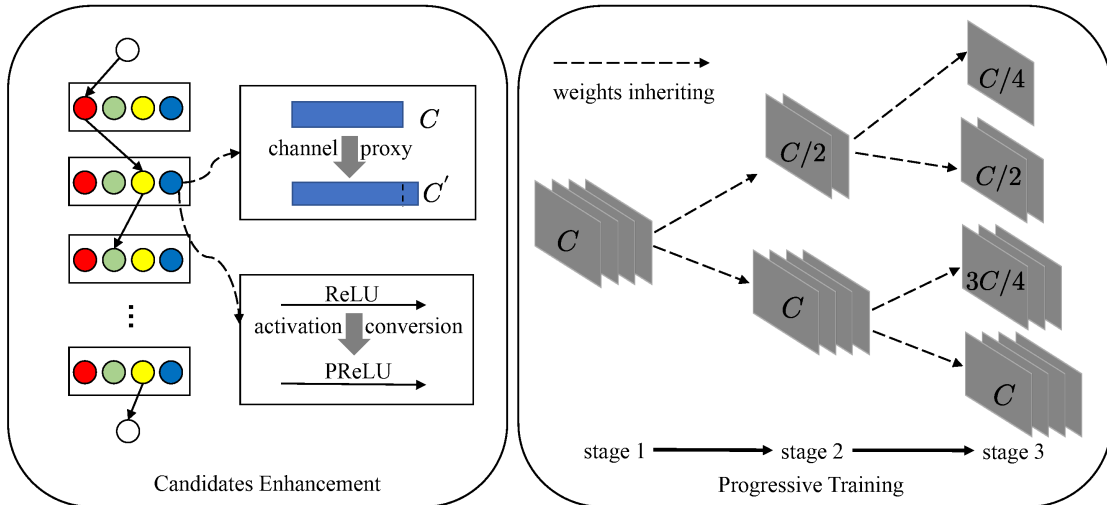


Figure 1. The framework of our method. The left part displays the candidates enhancement of supernet. The “blue” option is enhanced by channel proxy and activation conversion. The corresponding sub-networks are strengthened. The right part takes weights of one layer as example to show the progressive training of supernet. The weights are duplicated step by step for further finetuning

adds few weights but easy to train without extra hyper-parameters tuning. Besides, we utilize a progressive training method to alleviate the interference of sub-networks. For easily launching the training process, we firstly train the supernet by a variant of [1]. Then we gradually append narrow branches of each searchable layer and finetune the weights. The performance of our approach is certified in the *Supernet Track of CVPR2021 1st Lightweight NAS Challenge* and ranks the 1st place.

2. Method

2.1. Supernet Training with Distilling

We firstly introduce the algorithm of supernet training with knowledge distilling in our method. The search space is denoted as \mathcal{S} with weights θ . It contains sub-networks $\{s_i\}, i \in [1, N]$. N is the number of all sub-networks in the supernet. s_N represents the largest sub-network with maximum channels.

Supernet training with knowledge distilling can be formalized as

$$\theta^* = \arg \min_{\theta} \mathbb{E}_{s_i \sim \Gamma(\mathcal{S})} [\mathcal{L}(s_i, s_N, \theta)] \quad (1)$$

where θ^* denotes the optimized weights of supernet. \mathcal{L} is the optimization function. $\Gamma(\mathcal{S})$ is a sampling distribution of $s_i \in \mathcal{S}$.

We utilize cross entropy (CE) loss as a common optimization function for image classification task. Then, Kullback–Leibler (KL) divergence loss is additionally applied

for distilling sub-networks with the largest one. The complete optimization function is defined as follows:

$$\mathcal{L} = (1 - \alpha)\mathcal{L}_{CE}(\mathcal{N}(s_i, \theta)) + \alpha\mathcal{L}_{KL}(\mathcal{N}(s_i, \theta), \mathcal{N}(s_N, \theta)), \quad (2)$$

where \mathcal{L}_{CE} and \mathcal{L}_{KL} denote the cross entropy loss and Kullback–Leibler divergence loss respectively. $\mathcal{N}(s_i, \theta)$ represents the sub-network with architecture s_i and inherited weights from θ . α is a coefficient which trades off classification loss and distillation loss. The training algorithm is illustrated in Algorithm 1.

Search Space. The search space is built based on the ResNet20 backbone. The numbers of channels in each layer are searchable. The candidate channel options of layers are as follows:

- layers 1 to 7: [4, 8, 12, 16].
- layers 8 to 13: [4, 8, 12, 16, 20, 24, 28, 32].
- layers 14 to 19: [4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 64].

There are around $7.21 * 10^{16}$ sub-networks in total.

2.2. Candidate Enhancing

In order to alleviate the interference between sub-networks in the supernet, we redesign the sub-networks for capacity enhancing. We define a mapping function \mathcal{R} to execute the transformation $s_i \rightarrow \mathcal{R}(s_i)$.

In this paper, we redesign the sub-networks with two types of modification: activation conversion and options enhancement. The left part of Figure 1 shows details of the modification. Firstly, we argue that the activation function “ReLU” is harmful for the small channels, which will filter too much information while the channels are only 4 or 8. We change the activation function to “PReLU” which is more smooth. In addition, the sub-networks in our search space within numerous channel options of 4 or 8 exhibit much lower accuracy in the supernet compared with stand-alone training. This inspires us to enhance the capacity of channel options 4 and 8. We employ channel options 5 and 9 as proxies of options 4 and 8 respectively, which contribute to similar accuracy when they are trained stand alone. With the modifications, the accuracies of sub-networks are effectively increased and it makes a better accuracy ranking of sub-networks in the supernet.

2.3. Progressive Training

In the first stage of supernet training, we only maintain one copy of weights for different options in each layer as [1] does and train the supernet by Algorithm 1. Although this benefits the training of small sub-networks in the beginning, the mutual interference of sub-networks becomes more intense since the complete weight sharing.

Therefore, we insert extra counterparts of weights for each layer and allocate them to different searchable options, which leads the original supernet to a multi-branch one. The transformation and training of supernet are implemented step by step for elaborately supernet finetuning. To be specific, we duplicate the weights of layers in the supernet based on the pretrained weights of previous training stage. The options of layers are divided into two parts and inherit different counterparts of the weights, which is exhibited in the right part of Figure 1. Then, we add more copy of weights for each layer and gradually reduce the number of options that share the same counterpart of weights. With progressive training, sub-networks in the supernet can be elaborately finetuned, which leads to further improved accuracy with pretrained weights.

3. Experiments

We evaluate our method in the *Supernet Track of CVPR2021 1st Lightweight NAS Challenge*. We firstly train the largest sub-network for 300 epoches with batch size 128 and apply a stochastic gradient descent optimizer with a momentum of 0.9. The learning rate is decreased from an initial value of 0.1 to 0 with a cosine learning rate decay strategy. The weights are regularized with weight decay of $5e - 4$. The data augmentation includes transformation of brightness and contrast, rotation of 15 degrees and random flipping. We implement the progressive training with the

Algorithm 1 Supernet Training

Input: supernet \mathcal{S} with weights θ , training set D_{train} , sampling number of sub-networks K , training iterations T , loss function \mathcal{L} .

Output: optimized supernet weights θ^* .

- 1: random initialize θ ,
 - 2: train largest sub-network of supernet with \mathcal{L}_{CE} .
 - 3: **for** $t \leftarrow 1, T$ **do**
 - 4: set gradients of all weights to zeros;
 - 5: forward largest sub-network s_N ;
 - 6: calculate gradients based on \mathcal{L}_{CE} .
 - 7: **for** $i \leftarrow 1, K$ **do**
 - 8: Sample sub-network s_i by uniform sampling;
 - 9: forward sub-network s_i ;
 - 10: distill the outputs of s_i with the outputs of s_N ;
 - 11: calculate gradients based on \mathcal{L} .
 - 12: **end for**
 - 13: update θ by accumulated gradients.
 - 14: **end for**
 - 15: return θ^* .
-

same hyper-parameters of training as largest sub-network except a different learning rate.

Metrics. We follow the setting of CVPR2021 challenge and evaluate the ranking performance of supernet with the absolute value of the Pearson correlation coefficient. We test 50,000 sub-networks provided by the challenge with inherited weights of the well-trained supernet. The Pearson correlation coefficient is calculated with a part of the sub-networks.

3.1. Results of Candidates Enhancement

We evaluate the effectiveness of candidates enhancement with one-stage supernet training illustrated in Algorithm 1. The number of sub-networks sampling in each weight update step is set to 8 and the initial learning rate of supernet training is 0.01.

The experimental results are displayed in Table 1, where “base” represents the original supernet, “OE” means the supernet with options enhancement, “PRL_OE” denotes the supernet with both options enhancement and activation conversion to “PReLU”. From the table, we find that the ranking correlation of *ResNet20_SPN_OE* outperforms that of base supernet *ResNet20_SPN* by 0.006. The score is further increased to 0.97321 after combining options enhancement and activation conversion, indicating the availability of candidates enhancement in our method.

3.2. Results of Progressive Training

The progressive training is consisting of three stages. Firstly, we train the supernet with complete weight sharing, where all searchable options in one layer inherit the same

Model	Supernet	Pearson Coeff.
<i>ResNet20_SPN</i>	base	0.96341
<i>ResNet20_SPN_OE</i>	OE	0.96944
<i>ResNet20_SPN_PRL_OE</i>	PReLU+OE	0.97321

Table 1. The ranking correlation of sub-networks in the supernet. “OE” represents the supernet with options enhancement.

weights. Afterwards, the weights are duplicated and searchable options for each layer are divided into two sets, such as splitting [4,8,12,16] to [4,8] and [12,16]. Finally, the number of weights in the supernet is further increased. Each option will have an exclusive counterpart of weights. The structures of supernet and training process are displayed in the right part of Figure 1. The supernet training of second stage is implemented based on the pretrained weights of first stage with an initial learning rate of 0.001. Third stage utilizes the same learning rate as second stage with latest pretrained weights.

Table 2 exhibits the experimental results of progressive training. The table demonstrates that second stage training makes a great improvement compared with the result of stage 1 by 0.0039 in “base” supernet and 0.0033 in “PReLU+OE” supernet. However, the training of third stage only works in supernet “PReLU+OE” because of the lower capacity of “base” supernet.

Model	Supernet	Training stage	Pearson Coeff.
<i>ResNet20_SPN</i>	base	1	0.96341
<i>ResNet20_SPN</i>	base	2	0.96732
<i>ResNet20_SPN</i>	base	3	0.96686
<i>ResNet20_SPN_PRL_OE</i>	PReLU+OE	1	0.97321
<i>ResNet20_SPN_PRL_OE</i>	PReLU+OE	2	0.97648
<i>ResNet20_SPN_PRL_OE</i>	PReLU+OE	3	0.97696

Table 2. The ranking correlation of supernet in different progressive-training stages.

3.3. Visualization of ranking correlation

We train a few sub-networks independently and get their accuracies on validation set. Then the correlation of accuracies obtained with supernet and stand-alone training are displayed in Figure 2, which visualizes the improvement of accuracies and ranking correlation of sub-networks in the supernet.

4. Conclusion

In this paper, we improve the ranking correlation of supernet with capacity enlarging and progressive training pipeline. Firstly, We utilize simple yet effective channel proxies and activation conversion for candidates enhancing. Secondly, we gradually add the counterpart of weights for different searchable options and elaborately finetune the supernet, which contributes to better convergence and ranking

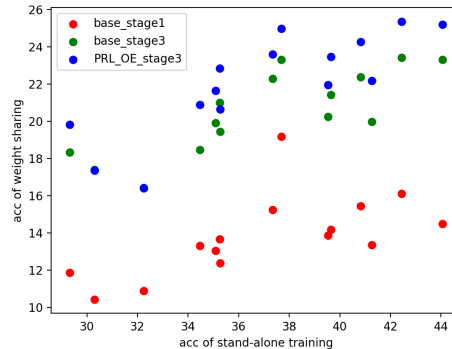


Figure 2. The accuracy correlation of sub-networks in different training settings.

consistency. Finally, the experiments demonstrate that both candidates enhancement and progressive training improve the ranking correlation of sub-networks in weights inheriting and stand-alone training.

References

- [1] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019. 1, 2, 3
- [2] Bo Chen, Golnaz Ghiasi, Hanxiao Liu, Tsung-Yi Lin, Dmitry Kalenichenko, Hartwig Adam, and Quoc V. Le. Mnasfpn: Learning latency-aware pyramid architecture for object detection on mobile devices. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1
- [3] Xiangxiang Chu, Bo Zhang, Jixiang Li, Qingyuan Li, and Ruijun Xu. Scarlet-nas: bridging the gap between stability and scalability in weight-sharing neural architecture search. *arXiv preprint arXiv:1908.06022*, 2019. 1
- [4] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. *arXiv preprint arXiv:1907.01845*, 2019. 1
- [5] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, pages 544–560. Springer, 2020. 1
- [6] Kevin Alexander Laube and Andreas Zell. Inter-choice dependent super-network weights. *arXiv preprint arXiv:2104.11522*, 2021. 1
- [7] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2820–2828, 2019. 1
- [8] Dongzhan Zhou, Xinchu Zhou, Wenwei Zhang, Chen Change Loy, Shuai Yi, Xuesen Zhang, and Wanli Ouyang. Econas:

Finding proxies for economical neural architecture search.
In *Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. [1](#)