One-Shot Neural Channel Search: What Works and What's Next

Chaoyu Guan, Yijian Qin, Zhikun Wei, Zeyang Zhang, Zizhao Zhang, Xin Wang, and Wenwu Zhu
Department of Computer Science and Technology, Tsinghua University
{guancy19,qinyj19,weizk19,zy-zhang20,zhang-zz18}@mails.tsinghua.edu.cn
{xin_wang,wwzhu}@tsinghua.edu.cn

Abstract

One-shot Neural Architecture Search (NAS) has been successful in discovering powerful architectures for a wide range of research applications. However, the supernets trained using one-shot methods suffer from the low-rank correlation problem, leading to sub-optimal or even failed architectures. To tackle the problem, we employ Neural Channel Search (NCS) to explore how to increase the consistency of supernet parameters in this paper. We first abstract a unified framework for one-shot NAS, then explore and compare recent advances on each part of the framework. We build a more powerful one-shot algorithm based on our abstracted framework, which further enforces the consistency of the supernet. Our approach wins 3rd place (tied) in the supernet track of the CVPR 2021 lightweight NAS competition. Extensive experiments based on our framework are conducted to demonstrate the efficacy of our proposed algorithm. We also summarize the recent breakthroughs and point out research directions deserving future investigations.

1. Introduction

Recent years have witnessed the success of Neural Architecture Search (NAS) on various kinds of applications such as Computer Vision [15, 7, 14], Natural Language Processing [9, 10, 4], Graph Representation Learning [13, 3], etc. Among which, supernet-based one-shot NAS, as a highly efficient performance estimation method, has been widely adopted to quickly derive models with satisfictory performances [5, 2, 11].

However, one-shot NAS, utilizing supernet to provide performance estimation for each model in the search space, is known to have low-rank correlation with the ground truth performance [12]. The low-rank correlation problem will result in biased rank prediction, thus providing sub-optimal or even failing architectures for the given task. Various methods [2, 11, 8] have been proposed to improve the consistency of the supernet when training. In this paper, we tackle this problem by utilizing a carefully designed combination of parameter sharing strategy, fair sampling strategy, tuned hyperparameters, and running-teacher-based knowledge distillation methods to further enhance the supernet consistency. Extensive comparison experiments are provided to validate the efficacy of our proposed approach. We also summarize recent breakthroughs and advances, together with the promising directions that deserve further exploration.

In particular, our method explores the recent best practices in the one-shot NAS literature. We first summarize the overall framework of one-shot NAS, each step of which has then been studied by exploring several existing methods. In detail, we explore and break common one-shot NAS framework into the following steps: (1) supernet construction (how to share weights of individual models), (2) architecture sampling, training data sampling (how to choose the data and architecture to be optimized), and (3) supernet optimization (how to define loss and update parameters of supernet). We employ Neural Channel Search (NCS) to justify the collected methods. Through extensive experiments, we find that using an ordinal shared supernet, distilling knowledge from the largest architecture to all sub-architectures, and carefully tuned hyperparameters can give supernets better consistency. A variation of our method wins 3rd place (tied) in the supernet track of CVPR 2021 lightweight NAS competition, with a 0.8324 Kendall score on the final dataset.

In summary, this paper makes the following contributions:

- We abstract current one-shot Neural Channel Search (NCS) algorithms into a unified framework, with publicly available libraries and recent advanced algorithms for each component[1].
- We propose a new algorithm for NCS, which achieves new state-of-the-art rank correlations between supernet and trained-from-scratch models.
- We provide extensive comparative studies to test the

[1] https://github.com/MetaLearners/CVPR2021-NAS-competition-Track-1-4th-solution

Layer	Channel Choices
1 - 7	4, 8, 12, 16
8 - 13	4, 8, 12, 16, 20, 24, 28, 32
14 - 19	4, 8, 12, 16, 20, 24, 28, 32 36, 40, 44, 48, 52, 56, 60, 64

Table 1. The search space of Lightweight NAS Challenge

effectiveness of recent advanced algorithms, summarizing the useful methods and tricks as well as providing insights for future research directions.

2. Lightweight NAS Challenge Supernet Track

In this section, we quickly overview the Supernet Track of CVPR 2021 Lightweight NAS Challenge. The supernet track aims at narrowing the performance gap between candidates with the parameters extracted from the shared parameters and the same architectures with the parameter trained independently. The challenge chooses NCS as the target problem and estimates the correlation of performance extracted from supernet and trained independently using Pearson or Kendall.

The search space is defined following ResNet with 20 layers (19 convolution layers + 1 FC layer). The selectable channel numbers are listed in Table 1, resulting in a space containing 1.15×10^{18} architectures. There are skip connections every two convolution layers since the 2nd layer. When the channel numbers do not match, the skip connection becomes a 1x1 convolution. There are also 2 downsampling layers at the 7th and the 13th convolutions. Our task is to estimate the accuracy of 50k predefined architectures randomly sampled from the search space using supernet based one-shot method. At the final stage, a subset of the given 50k architectures will be used to evaluate the trained supernet using Kendall-Tau as the metric.

3. Unified Framework For One-shot NCS

3.1. One-shot Formulation

An NCS problem can be defined as a bi-level optimization problem [7]:

$$\begin{aligned} a^* &= \text{argmin}_{a \in A} \, L_{val}(a, \mathbf{w}^*(a)), \\ \text{s.t. } \mathbf{w}^*(a) &= \text{argmin}_{\mathbf{w} \in \mathbf{W}(a)} \, L_{train}(a, \mathbf{w}), \end{aligned} \tag{1}$$

where A represents the search space stated in Table 1, $\mathbf{W}(a)$ represents the parameter space for architecture a, $\mathbf{w}^*(a)$ is the best parameters of architecture a, and a^* is the global optimal solution of NCS. Directly deriving the best architecture needs training all the models in search space, which is not applicable since the search space is always too large to traverse. To solve the problems above more efficiently, one way is to share the parameters of all models and jointly optimize them altogether [5], which is called one-shot based methods. The optimization problem then becomes:

$$\begin{aligned} a^* &= \text{argmin}_{a \in A} \, L_{val}(a, \mathbf{w}^*), \\ \text{s.t. } \mathbf{w}^* &= \text{argmin}_{\mathbf{w} \in \mathbf{W}} \, \mathbb{E}_{a \sim \Gamma(A)} L_{train}(a, \mathbf{w}), \end{aligned} \tag{2}$$

where $\mathbf{w} \in \mathbf{W}$ is the shared parameter of supernet, and $\Gamma(A)$ is the distribution of architectures in search space. The major challenge in this problem formulation is how to train the shared weights so that it can provide reliable architecture rank information for architecture search, which is a sufficient condition to find the global optimal architecture.

We divide the training process of the supernet into several steps. The Supernet Construction step aims at building a supernet that contains all the models in search space. The Architecture and Data Sampling step choose the architecture and data to optimize in the constructed supernet. Then, the corresponding parameters are optimized in the Supernet Optimization step. We evaluate the negative cross-entropy loss of each architecture using the parameters in supernet to represent its performance and report the Kendall Tau of the performance derived from supernet with the Top-1 Accuracy of trained individual models.

3.2. Supernet Construction

There are in total three kinds of layers to be considered: Conv, BN, and FC. How to share their parameters properly is the key problem when constructing supernet.

For convolution layer, we can consider their input channel number and output channel number. We explore four share strategies:

- **Independent** Each convolution choice use separate parameters.
- **Front** The convolutions with the same input channel number share the same super parameters.
- **End** The convolutions with the same output channel number share the same super parameters.
- **Full** All convolutions share the same super parameters.

Correspondingly, there are also four kinds of strategies to share parameters of batch normalization. The sharing and deriving strategies are the same as convolutions.

For fully connected layers, since the output dimension is always 100, we explore two kinds of strategies:

- **Independent** Each FC use separate parameters.
- **Full** All FC use the same parameters.

3.3. Architecture and Data Sampling

After deciding how to construct the supernet, we need to determine how to select the architectures and data to optimize it. For data batch sampling, we simply use random shuffle and sample strategy to stay consistent when training each model. For architecture sampling, we consider the following strategies:

- **Uniform** Uniformly sample each channel from the channel list [5].

Name	SPOS[5]	FairNAS[2]	AutoSlim[11]	Ours
K-Tau	0.7655	0.7718	0.7787	**0.8304**

Table 2. Comparison results of our best settings with previous state of the arts. K-Tau means kendall tau. The higher the better.

- **Fair** Enforce each channel choice appear exactly once every L architecture samples [2], where L is the size of channel list.

Besides, there are also two ways to align data with architectures. One can sample and calculate the loss of several architectures over the same data batch, or assign each architecture a different data batch for estimating the expected loss over the whole dataset. We use `same-batch (sb)` to represent whether to calculate sampled architectures over the same data batch.

3.4. Supernet Optimization

In this section, we explore how to optimize the supernet parameters given chosen architectures and data batches. The simplest way is to directly minimize the cross-entropy loss between supernet and ground truth labels [5, 2]. Recently, some related works [8, 6, 1] use knowledge distillation instead of cross-entropy to help lower the difficulty of training supernet, which lead to a supernet with better consistency and performance.

In this paper, we explore the following optimization methods.

- **Naive** Directly use cross-entropy loss [5].
- **Fixed Teacher** First train the biggest architectures in the space, then use it to teach each architecture [1].
- **Running Teacher** The teacher is trained from scratch with the students at the same iteration [11].

4. Experiment

In this section, we show extensive experiments on every part of the proposed framework. We first show the performance of the best settings we've got on the space published in Challenge, then ablate on each part of the proposed framework to show some insights and conclusions.

4.1. Best Settings

The best supernet training settings are: FullConv, EndBN, and IndependentFC for supernet building, fair sampling with architecture batch as 18 and use same batch strategy, and running-teacher knowledge distillation. We compare our best settings with the previous state-of-the-art supernet training methods. The results are shown in Table 2. We can see that the proposed methods significantly improve the rank correlations.

We then ablate each part of the proposed framework. For efficiency reasons, unless specially stated, we use FullConv, FullBN, FullFC, Fair Sampling with architecture batch size $n = 1$, and Naive Optimization as the default settings.

Conv.	BN	FC	K-Tau
Full	Full	Full	0.7718
Front			0.4306
End			0.5137
Independent			0.3760
	Front		0.7899
	End		0.7829
	Independent		0.7823
		Independent	0.7895
Front	Front		0.4764
End	End		0.4638
Independent	Independent		0.2867
	Front	Independent	0.7875
	End	**Independent**	**0.7995**
	Independent	Independent	0.7813

Table 3. Ablation on constructing supernet with different sharing strategies.

Sample	Uniform	Fair
K-Tau	0.7655	**0.7718**

Table 4. Ablation on different architecture sampling strategies

4.2. Ablation on Supernet Construction

We explore the combination of supernet constructions on how to share different components. We start from the supernet with the biggest share ratio (FullConv, FullBN, FullFC), and replace each component with other share settings. We also test the combinations of the same share strategies on Conv and BN. The results are shown in the upper part of Table 3. One can observe that for convolution, the fully shared version performs best, with a large superior over other kinds of share strategies. However, the BN and FC give different conclusions. The reason may be that the convolution has too many parameters to train, thus need higher share strategies to converge better. While the less parameterized part like BN and FC should have lower share levels to customize on each individual model.

We thus explore further to combine less shared BNs with IndependentFC and report their performances in the lower part of Table 3. We can see that EndBN combined with IndependentFC gives the best performance. We use this combination as the final solution for supernet construction.

4.3. Ablation on Data Sampling

We show the comparison results of different sampling strategies in Table 4. We observe that the fair sampling strategy is slightly better than uniform. We also test the number of samples for each parameter update, and test the effectiveness of the same batch strategy. The results are shown in Figure 5. We can see that as the architecture number increase, the correlation first increases and then decreases,

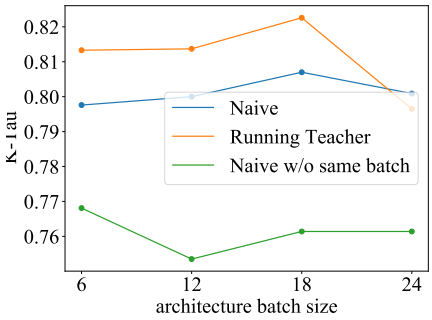

Figure 1. Comparison of different supernet optimization methods under different architecture batch size.

Type	Naive	Fixed Teacher
K-Tau	**0.7718**	0.7657

Table 5. Comparison of different supernet optimization methods under the setting of sampling architecture batch $n = 1$.

with $n = 18$ as the best value. The same batch strategy can also improve consistency.

4.4. Ablation on Supernet Optimization

We first compare the naive and fixed teacher optimization methods using architecture sample batch size $n = 1$. The results are shown in Table 5. We can see that the naive optimization strategy is better than the fixed teacher strategy. We then compare the naive strategy and running teacher strategy under different ns in Figure 5. Running teacher strategy also achieves the best consistency using $n = 18$. We use running teacher for the final best settings since its best K-Tau outperforms the naive method.

4.5. Future Directions

Apart from the comparative studies above, we also find some interesting patterns and phenomenons that inspire practical future works.

Sharing Strategies We find that different sharing levels of Conv, BN, and FC show different influences on the consistency of supernet. One may further enhance the consistency by sharing the parameters more wisely, like mixed-level share strategies or multi-stage train and share strategies.

Catastrophic Forgetting We find that even using the fair sample strategy, the rank correlation is still not stable across different training epochs. The reason is probably that the architecture space is too large, and the supernet tends to forget the trained architectures earlier. Avoiding forgetting earlier architectures may be beneficial to the consistency problem.

Balance of Parameter Size Actually, the parameter size has a mild correlation with the final performances. The Kendall Tau is around 0.5. One may further explore how to let larger architectures trained better to leverage the parameter size prior.

5. Conclusions and Future Works

In this paper, we abstract a unified framework for supernet-based one-shot NAS. We explore and compare several recent advancements for each component of the proposed framework, and combine the useful strategies and tricks to further enhance the consistency. We also provide extensive ablation studies and summarize the promising future directions for the supernet consistency problem.

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. In *ICLR*, 2020. 3

[2] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. *CoRR*, abs/1907.01845, 2019. 1, 3

[3] Yang Gao, Hong Yang, Peng Zhang, Chuan Zhou, and Yue Hu. Graph neural architecture search. In *IJCAI*, 2020. 1

[4] Chaoyu Guan, Xin Wang, and Wenwu Zhu. Autoattend: Automated attention representation search. In *ICML*, 2021. 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 *ECCV*, 2020. 1, 2, 3

[6] Changlin Li, Jiefeng Peng, Liuchun Yuan, Guangrun Wang, Xiaodan Liang, Liang Lin, and Xiaojun Chang. Block-wisely supervised neural architecture search with knowledge distillation. In *CVPR*, 2020. 3

[7] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *ICLR*, 2019. 1, 2

[8] Houwen Peng, Hao Du, Hongyuan Yu, Qi Li, Jing Liao, and Jianlong Fu. Cream of the crop: Distilling prioritized paths for one-shot neural architecture search. In *NeurIPS*, 2020. 1, 3

[9] David R. So, Quoc V. Le, and Chen Liang. The evolved transformer. In *ICML*, 2019. 1

[10] Yujing Wang, Yaming Yang, Yiren Chen, Jing Bai, Ce Zhang, Guinan Su, Xiaoyu Kou, Yunhai Tong, Mao Yang, and Lidong Zhou. Textnas: A neural architecture search space tailored for text representation. In *AAAI*, 2020. 1

[11] Jiahui Yu and Thomas S. Huang. Universally slimmable networks and improved training techniques. In *ICCV*, 2019. 1, 3

[12] Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. In *ICLR*, 2020. 1

[13] Huan Zhao, Lanning Wei, and Quanming Yao. Simplifying architecture search for graph neural network. In *CIKM workshop*, 2020. 1

[14] Hongpeng Zhou, Minghao Yang, Jun Wang, and Wei Pan. Bayesnas: A bayesian approach for neural architecture search. In *ICML*, 2019. 1

[15] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017. 1