FTSO: EFFECTIVE NAS VIA FIRST TOPOLOGY SECOND OPERATOR

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing one-shot neural architecture search (NAS) methods generally contain a giant supernet, which leads to heavy computational costs. Our method, named FTSO, separates the whole architecture search into two sub-steps. In the first step, we only search for the topology, and in the second step, we only search for the operators. FTSO not only reduces NAS's search time from days to 0.68 seconds, but also significantly improves the accuracy. Specifically, our experiments on ImageNet show that within merely 18 seconds, FTSO can achieve 76.4% testing accuracy, 1.5% higher than the baseline, PC-DARTS. In addition, FTSO can reach 97.77% testing accuracy, 0.27% higher than the baseline, with 99.8% of search time saved on CIFAR10.

1 INTRODUCTION

Since the great success of AlexNet (Krizhevsky et al., 2012) in image classification, most modern machine learning models have been based on deep neural networks. For neural networks, their performance is greatly determined by the architectures. Thus, in the past decade, a tremendous amount of work (Simonyan & Zisserman, 2015; Szegedy et al., 2015; He et al., 2016) has been done to investigate proper network architecture design. However, as the network size has grown larger and larger, it has gradually become unaffordable to manually search for better network architectures via trial and error due to the heavy time and resource overhead. To ease this problem, a new technique called neural architecture search (NAS) was introduced, which allows computers to search for better network architectures automatically instead of leaving these workloads to humans.

Early-proposed reinforcement learning-based NAS methods (Zoph & Le, 2017; Baker et al., 2017; Zoph et al., 2018) typically have an RNN-based controller to sample candidate network architectures from the search space. These sampled architectures are separately trained from scratch in order to obtain accuracy on the target dataset. Then the accuracy is further utilized as rewards to update the controller's parameters. Although these algorithms can provide promising accuracy, their computational costs are usually unaffordable, e.g., 1800 GPU-days are required for NASNet.

To ease the search efficiency problem, one-shot approaches (Pham et al., 2018; Cai et al., 2019; Liu et al., 2019), with parameter sharing, were proposed. It first creates a huge directed acyclic graph (DAG) supernet, containing the whole search space, and then shares the kernel weights among all the sampled architectures via the supernet. Although one-shot methods do not need to repeatedly retrain the sampled networks from scratch, they suffer heavily from their supernets' computational overheads, which is particularly severe in differentiable models (Liu et al., 2019; Xu et al., 2020).

Limited by current NAS algorithms' inefficiency, it is still rather challenging to find satisfying network architectures on large-scale datasets and high-level tasks. For instance, current speed-oriented NAS approaches generally require days to accomplish one search trial on ImageNet, e.g., 8.3 GPU-days for ProxylessNAS (Cai et al., 2019) and 3.8 GPU-days for PC-DARTS (Xu et al., 2020). The huge search cost prevents NAS's further development and application in both academia and industry. To ease these problems, we argue that it is a must to propose a new well-defined search space, which is both expressive enough to cover the most powerful architectures and compact enough to filter out the poor ones. However, the discovery on the search space heavily relies on human's prior knowledge, and blindly pruning the search space might lead to missing the best architectures. We also notice that the correlation among different architectures is not well investigated in existing works. This result leads to the lack of prior knowledge for effectively shrinking the search space.

Motivated by Shu et al. (2020), who demonstrated that randomly replacing operators in a found architecture does not harm the accuracy much, we believe that omitting the influence of operators and clustering architectures according to the topology, may bring no reduction to the testing accuracy, while significantly benefiting the search efficiency. Thus, in this paper, we propose to separately search for the network topology and the operators. We name this new method Effective NAS via First Topology Second Operator (FTSO).

We first mathematically prove that, by greatly shrinking the graph of the search space, reducing the operators' complexity in magnitude, lowering the required searching period from 50 epochs to one iteration and significantly easing the Matthew effect, namely that the complex operators may never get the chance to be well tuned, thus the found architecture only contains very simple operators, and performs poorly on the testing set, FTSO reduces the required parameters by a factor of 0.53×10^8, decreases the FLOPs per iteration by a factor of 2×10^5 and significantly promotes the accuracy compared to the baseline, PC-DARTS. Then we empirically show that FTSO is superior in both efficiency and effectiveness, accomplishing the whole architecture search in 0.68 seconds. On ImageNet, FTSO can achieve 76.4% testing accuracy, 1.5% higher than the baseline, PC-DARTS, within merely 18 seconds. What is more, when we only search for one iteration, FTSO can reach 75.64% testing accuracy, 0.74% higher than the baseline in just 0.68 seconds. Besides, if we allow FTSO to search for 19 minutes, 76.42% Top1 and 93.2% Top5 testing accuracy can be obtained. In addition, FTSO can reach 97.77% testing accuracy, 0.27% higher than the baseline, with 99.8% of search time saved on CIFAR10. Although in this paper we only implement FTSO within a continuous search space, in fact, it can easily be transferred and bring improvements to other NAS algorithms.

2 Related Work

Deep neural networks have won great success and been widely adopted as the basic framework of modern machine learning algorithms. However, due to the continuing growth of the size and the complexity of neural networks (He et al., 2016; Devlin et al., 2019; Radford et al., 2019; Brown et al., 2020), it has become harder and harder to manually design new network structures. To deal with these problems, a new technique, named neural architecture search (NAS), was proposed to allow computers design the network architectures automatically. In general, existing NAS algorithms can be divided into three categories, namely, reinforcement learning-based, revolution-based and differentiable.

Early-proposed reinforcement learning-based methods (Zoph & Le, 2017; Zoph et al., 2018) generally suffer from high computational cost. ENAS (Pham et al., 2018) reduces the time cost by forming an over-parameterized network (a.k.a. supernet) containing all the possible architectures. Every time the controller picks one sub-network from the supernet, all the matched weights are inherited. At the same time, all the updates on the picked sub-network will be equally imposed on the supernet.

Although reinforcement learning-based NAS methods perform well in terms of accuracy, they suffer from low-efficiency sampling. Instead of sampling a discrete architecture and then evaluating it, DARTS (Liu et al., 2019) treats the whole search space as a continuous supernet. It assigns every operator a real number weight and treats every node as the linear combination of all its transformed predecessors. To be specific, DARTS's search space is a directed acyclic graph (DAG) containing two input nodes inherited from previous cells, four intermediate nodes and one output node. Each node denotes one latent representation and each edge denotes an operator. Every intermediate node $\mathbf{x}_j$ is calculated from all its predecessors $\mathbf{x}_i$, i.e., $\mathbf{x}_j = \sum_{i<j} \sum_{o \in \mathcal{O}} \frac{\exp \alpha_{i,j}^o}{\sum_{o' \in \mathcal{O}} \exp \alpha_{i,j}^{o'}} o(\mathbf{x}_i)$, where $\mathcal{O}$ denotes the collection of all candidate operators, $\alpha_{i,j}^o$ denotes the weight for operator o from node i to j. This strategy allows DARTS to directly use gradients to optimize the whole supernet. After the supernet converges, DARTS only retains the operators with the largest weights. In this way, the final discrete architecture is derived. The main defect of DARTS is that it needs to maintain and do all calculations on a giant supernet, which inevitably leads to heavy computational overheads and over-fitting.

Proposed to relieve the computational overhead of DARTS, DARTS-ES (Zela et al., 2020) reduces the number of searching epochs via early stopping, according to the Hessian matrix's max eigen-

value. PC-DARTS (Xu et al., 2020) decreases the FLOPs per iteration by only calculating a proportion of the input channels and retain the remainder unchanged, and normalizes the edge weights to stabilize the search. To be specific, in PC-DARTS, every intermediate node $\mathbf{x}_j$ is computed from all its predecessors $\mathbf{x}_i$, i.e., $\mathbf{x}_j = \sum_{i<j} \frac{\exp \beta_{i,j}}{\sum_{i'<j} \exp \beta_{i',j}} f_{i,j}(\mathbf{x}_i)$, where $\beta_{i,j}$ describes the input node i's importance to the node j, and $f_{i,j}$ is the weighted sum of all the candidate operators' outputs between node i and j. Specifically, $f_{i,j}(\mathbf{x}_i, \mathbf{S}_{i,j}) = \sum_{o \in \mathcal{O}} \frac{e^{\alpha^o_{i,j}}}{\sum_{o' \in \mathcal{O}} e^{\alpha^{o'}_{i,j}}} o(\mathbf{S}_{i,j} * \mathbf{x}_i) + (1 - \mathbf{S}_{i,j}) * \mathbf{x}_i$, where $\mathbf{S}_{i,j}$ denotes a binary vector, in which only $1/K$ elements are 1.

Recently, there are more and more papers realizing the importance of the network topology. For example, Gaier & Ha (2019) tries to find a network topology irrelevant to detailed weights. It starts from sparse and simple topologies, then assigns these topologies a range of weights, and directly evaluates these architectures without further tuning. After these, it picks out the architectures with the highest evaluation metrics, and varies them to produce the new candidate topologies. Different from Gaier & Ha (2019), AutoShrink (Zhang et al., 2019) starts from a full DAG. In every loop, it first creates a set of new graphs with one edge fewer than the previous found DAG, then randomly picks some generated DAGs out to form new architectures, which are then evaluated to obtain the metrics. Finally, the architecture with the highest evaluation metric is sent to the next loop.

3 FTSO: Effective NAS via First Topology Second Operator

Existing NAS approaches generally suffer from the heavy computational overhead and the unsatisfying testing accuracy leaded by the huge search space. Such problems are especially stern in one-shot and differentiable methods because, these algorithms need to maintain and even do all the calculations directly on the search space.

To ease such problems, it is in a great demand to investigate the correlations among different architectures and to shrink the search space according to these prior knowledge. We notice that there is an important observation in Shu et al. (2020) that randomly substituting the operators, in a found architecture, does not observably influence the testing accuracy. This gives us an inspiration that it might be possible to cluster the architectures according to their connection topologies. Thus, in this paper, we propose to first search for the network topology with simple operators, and then fix the topology, and search for the most suitable operators for the given topology. In this way, the testing accuracy can still be guaranteed, while the search space is shrunken in magnitude. We name this new NAS algorithm Effective NAS via First Topology Second Operator (FTSO).

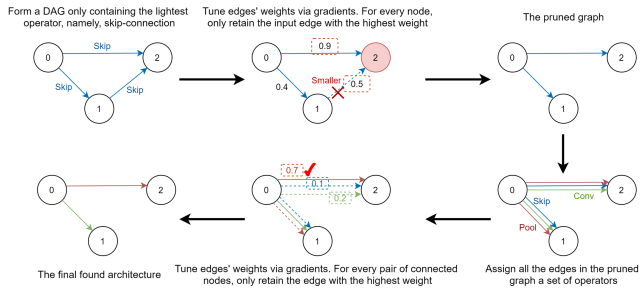


Figure 1: The main structure of FTSO.

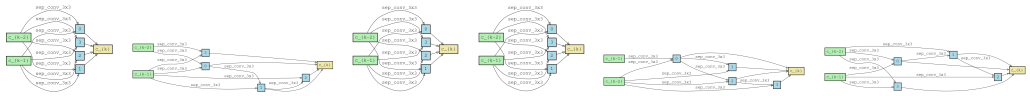
As shown in Figure 1, we inherit the differentiable framework of PC-DARTS, and divide the architecture search into two phases. In the first phase, we use the simplest operator, *skip connection*, to replace all the candidate operators. Because the *skip connection* operator contains no kernel weights and there is merely one candidate operator, we only need to optimize the architecture parameters $\beta_{i,j}$, denoting the node i's importance to the node j. After the topology search, for every intermediate node j, we only retain its connections to its predecessors i^* with the highest two $\beta_{i,j}$. In the second phase, we search for the operators suitable for the found topology. To be specific, here we have two solutions. The first strategy is to first replace all the operators, in the found topology, with mix-operators $f_{i,j}$. Here $f_{i,j}$ is the linear combination of all candidate operators o, with weight $\alpha^o_{i,j}$,

which denotes the importance of the operator o, between the node i and j. Then we optimize the architecture parameters, $\alpha^o_{i,j}$ and $\beta_{i,j}$, and the kernel weights $\omega^o_{i,j}$ alternatively. After the operator search, for every two connected nodes i and j, we only retain one operator o^* with the highest $\alpha^o_{i,j}$. The second strategy is to directly replace all the operators, in the found topology, with one single operator, which owns the greatest potential after fully converging, e.g., a convolution operator. In this paper, we take the second strategy as the default configuration because, it is much more efficient, and the large amount of kernel weights, contained in the first strategy, may lead to over-fitting, and finally leads to worse performance than the second strategy. Additional empirical results can be found in Section 4.3.

In DARTS, the network topology and operators are searched jointly, which makes both the size and computational cost of the supernet extremely high because, the total number of operators in DARTS is $\frac{1}{2}(n^2-3n)p$. Here n denotes the number of nodes and p denotes the number of candidate operators. By comparison, there are only $\frac{1}{2}n(n-3)$ operations in our topology search, and $2(n-3)p$ operations in our operator search. Since n is usually close to p, FTSO reduces the number of operations from $O(n^3)$ to $O(n^2)$.

In addition to the reduction on the number of operations, FTSO also dramatically decreases the internal cost of the operations because, all the complex operators are replaced with the simple operators, during the topology search. To be specific, a vanilla convolutional operator needs $k^2C_{in}H_{out}W_{out}C_{out}$ FLOPs and $(k^2C_{in}+1)C_{out}$ parameters, where k is the kernel size, C_{in} is the input tensor's channel number and H_{out}, W_{out}, and C_{out} are the output tensor's height, width and channel number respectively. By comparison, a *skip connection* operator needs 0 parameters and 0 FLOPs. As a typical configuration when $k=5$, $C_{in}=C_{out}=512$, $n=7$ and $p=8$, assume all the replaced operators are convolutions. Our algorithm requires only $\frac{1}{((k^2C_{in}+1)C_{out}+1)p}=1.9\times10^{-8}$ times the parameters and $\frac{n+2}{2np(k^2C_{in}+1)+4}=6\times10^{-6}$ times the forward-propagation FLOPs compared to DARTS.

The huge reduction on the parameter numbers provides us a large amount of benefits. Firstly. It allows the algorithm to converge within extremely few iterations. Suppose the batch size is 256, and we only search for one iteration on CIFAR10, then FTSO requires merely 10^{-10} times of the FLOPs, in the whole search procedure, compared to the baseline. Note that because our algorithm does not need to back-propagate with respect to the kernel weights, its actual superiority on efficiency is even more significant than these theoretical calculations. Secondly. Because our supernet contains very few parameters, it can hardly over-fit the dataset. Thus, the found architecture may generalize better on new datasets. Thirdly. In DARTS, every architecture is assumed to be able to converge in one iteration. Unfortunately, it is almost impossible for those complex supernets to converge in such a short period. As if this hypothesis does not stand, the whole theoretical foundation of DARTS may collapse. In FTSO, this problem is almost nonexistent because, the *skip connection* has no kernel weights to tune. Thus, the validation accuracy, obtained during the one-shot search, is exactly the real performance of the supernet, and the disadvantages brought by the bi-level optimization can be eased.



(a) (b) (c) (d) (e) (f)

Figure 2: FTSO's found architectures. (a) and (b): Normal and reduction cells found on CIFAR10 after one epoch's search; (c) and (d): Normal and reduction cells found on the entire ImageNet after one epoch's search; (e) and (f): Normal and reduction cells found on CIFAR10, where we adopt the operator search, and use the 3×3 *separable convolution* to search for the topology.

4 EXPERIMENTS

Our search algorithm is evaluated on the two most widely-used datasets in NAS papers, namely, CIFAR10 and ImageNet. Following DARTS, our search space contains a total eight candidate operators: 3×3 and 5×5 *separable convolutions*, 3×3 and 5×5 *dilated separable convolutions*, 3×3

max and *average pooling*, *skip connection* (i.e., $output = input$) and *zero* (i.e., $output = 0$). When searching for the topology, we pick only one operator from the candidate set. There are two possibilities: if the chosen operator contains no trainable kernel weights, we only optimize the architecture parameters $\beta_{i,j}$; otherwise, we optimize the kernel weights $\omega^o_{i,j}$ and the architecture parameters $\beta_{i,j}$ alternatively. When searching for the operators, we fix the pruned topology and construct a new supernet containing all the eight candidate operators. Then, we optimize both the kernel weights and the architecture parameters for 35 epochs after 15 epochs' pretraining (Chen et al., 2019) on the kernel weights. After the topology search, it is both okay to search for the topology with gradients or directly replacing the *skip connections* with other operators. In Section 4.1 and 4.2, we focus more on the second configuration because it is more efficient, and in Section 4.3, we have a comprehensive comparison on these two strategies. All our search experiments are done under the same 11GB VRAM constraint on one RTX 2080Ti GPU. The detailed configurations are shown in Table 1. In addition, our most experiments only search for one epoch or one iteration because, fewer epochs not only shorten the search time, but also avoid the over-fitting and provide higher generalization ability. For more experimental supports, please refer to Section 4.3. Note that it is almost impossible for the existing models to obtain satisfying results via only searching for one epoch or one iteration because, their supernets contain a large amount of parameters, which require a long period to tune.

Table 1: FTSO's searching configurations on CIFAR10 and ImageNet

Hyper-parameter	**Search Config.**		**Evaluation Config.**	
	CIFAR10	**ImageNet**	**CIFAR10**	**ImageNet**
Cells (Normal + Reduction)	$6+2$	$6+2$	$18+2$	$12+2$
Channel proportion	25%	25%	-	-
Initial channel number	16	16	36	48
Kernel optimizer	SGDM	SGDM	SGDM	SGDM
Kernel learning rate	Cos. (0.025, 0)	0.5	Cos. (0.025, 0)	Linear (0.5, 0)
Architecture optimizer	Adam	Adam	-	-
Architecture learning rate	6×10^{-4}	6×10^{-3}	-	-

* Some items are '-' because, when evaluating, no architecture parameters need to be updated.

4.1 Results on CIFAR10

Following DARTS, we search for the architecture in a cell manner, where each cell has four intermediate nodes and 16 initial channels during the search. We use half the images (25K) from the training set for kernel weight updating and the other half for architecture parameter updating. When evaluating, we stack 18 normal and two reduction cells to form the entire network.

We compare FTSO to existing state-of-the-art NAS methods in Table 2. The results show that FTSO can greatly surpass the baseline, i.e., PC-DARTS, by 0.27% in terms of the test accuracy on CIFAR10, with 99.8% of the search time saved when only searching for the topology with skip-connections and then replacing them all with 3×3 *separable convolutions*. The reason why we do not adopt 5×5 *separable convolutions* is that the pre-processed input images do not have enough resolutions, and the network is rather deep. After a few layers, the convolution's receptive field has been larger than the whole image. At that time, larger convolutional kernels may not bring benefits. Instead, the extra parameters brought by the larger kernel size may lead to over-fitting. Of course, suppose both the image's resolution and the dataset's scale are big enough, and the evaluation period is adequate, then the 5×5 *separable convolution* might be a better choice. The found architecture after one epoch's search is shown in Figure 2. Because FTSO contains very few trainable parameters, it can even achieve comparable accuracy to PC-DARTS with only an one-time gradient update. Under this configuration, merely 0.68 seconds are required and 99.993% of the search time is saved. In addition, as shown in Table 3, when the topology is searched with complex operators for a long period, additional operator search usually helps. However, when we search for the topology with simple operators for a short period, omitting the operator search may lead to better results. This is because with simple operators and very few updates, the found topology can already generalize quite well. In contrast, the supernet, during the operator search, contains a large number of parameters, so it tends to pick out architectures which converge more easily, while this kind of architectures usually contain simpler operators and lead to the lower final accuracy after fully converging.

4.2 RESULTS ON IMAGENET

On ImageNet, we use similar configurations to those on CIFAR10. However, both the number of images and the size of each image in ImageNet are significantly larger than those in CIFAR10. Thus we first reduce the images' resolution from 224×224 to 28×28 with three stride-2 convolutions, and when searching, we have two configurations. In the first configuration, following PC-DARTS, we only pick 10% and 2.5% of images from the training set for the kernel weights and architecture parameters' optimizations respectively; in the second configuration, thanks to FTSO's huge computational cost reduction, it is feasible for us to search with the whole of ImageNet. For evaluation, we stack 12 normal cells and two reduction cells to form the entire network and train it from scratch for 250 epochs. The detailed configurations are shown in Table 1.

Table 2: Comparison with existing state-of-the-art image classification architectures

Architecture	**CIFAR Err.(%)**		**ImageNet Err.(%)**		**Search Cost (GPU-days)**	
	600	**1200**	**Top1**	**Top5**	**CIFAR**	**I.N.**
NASNet-A†(Zoph et al., 2018)	2.65	-	26.0	8.4	1800	-
AmoebaNet†(Real et al., 2019)	2.55	-	24.3	7.6	3150	-
PNAS(Liu et al., 2018)	3.41	-	25.8	8.1	225	-
ENAS†(Pham et al., 2018)	2.89	-	-	-	0.5	-
DARTS (2nd)†(Liu et al., 2019)	2.76	-	26.7	8.7	1	4.0
SNAS†(Xie et al., 2019)	2.85	-	27.3	9.2	1.5	
ProxylessNAS$^{\dagger *}$(Cai et al., 2019)	2.08	-	24.9	7.5	4.0	8.3
P-DARTS†(Chen et al., 2019)	2.50	-	24.4	7.4	0.3	-
BayesNAS†(Zhou et al., 2019)	2.81	-	26.5	8.9	0.2	-
PC-DARTS(CIF.)†(Xu et al., 2020)	2.57	2.50	25.1	7.8	0.1	-
PC-DARTS(I.N.)$^{\dagger *}$(Xu et al., 2020)	-	-	24.2	7.3	-	3.8
FTSO (CIFAR10 + 1 epoch)†	2.48	**2.23**	23.60	7.01	2×10^{-4}	-
FTSO (CIFAR10 + 1 iteration)†	2.68	2.54	24.36	7.27	$\mathbf{7.87 \times 10^{-6}}$	-
FTSO (Full ImageNet + 1epoch)$^{\dagger *}$	**2.35**	2.26	**23.58**	**6.80**	-	**0.01**

† When testing on CIFAR10, these models adopt cut-out.
* These models are directly searched on ImageNet.

Our experiments in Table 2 show that FTSO is significantly superior to existing methods in both efficiency and effectiveness. Our model transferred from CIFAR10, with only one epoch's search, can achieve as high as 76.4% testing accuracy on ImageNet, 0.6% higher than the baseline's result directly searched on ImageNet, while saving 99.995% search time. Compared with the baseline's result transferred from CIFAR10, FTSO can achieve 1.5% accuracy improvement, with 99.98% search time saved. In addition, even when we only search for one iteration on the proxy dataset, our method can still surpass the baseline's result directly searched on ImageNet by 0.03%, in terms of the Top5 accuracy, whithin merely 0.68 seconds. Furthermore, when directly searching on ImageNet, 76.42% Top 1 and 93.20% Top 5 accuracy can be obtained. The found architectures after one epoch's search on CIFAR10 and the entire ImageNet are shown in Figure 2 . It is surprising that the best architecture we found on ImageNet is the shallowest and widest one. Compared to the much more "reasonable" architecture shown in Figure 2(e) and 2(f), which was found with the topology search only containing 3×3 separable convolutions and an additional operator search on CIFAR10, the "abnormal" architecture, containing the same amount of FLOPs and parameters, can achieve 0.78% higher testing accuracy. We think the reason is that, although deeper architectures have the larger model capacity, their loss curve might not be flat (Shu et al., 2020). Thus, they may not generalize as well as the shallow models we found.

4.3 ABLATION STUDY

We visualize the relationship between FTSO's testing accuracy and the number of searching epochs in Figure 3(a) and Figure 3(b). In terms of topology-only search, one epoch is just enough, thanks

to the many fewer kernel weights contained in FTSO, and more search epochs bring obvious disadvantages because of over-fitting. Since one epoch performs better than more epochs, it raises the question whether one iteration is also superior to more iterations. In Figure 3(c) we show that although one iteration cannot surpass one epoch, it is better than a few iterations because, in the latter configuration, only a small proportion of the training set is used, which brings variance to the model and hurts its generalization ability. In terms of whether we should search for one iteration or two, in Figure 3(d), we show that both choices work well.

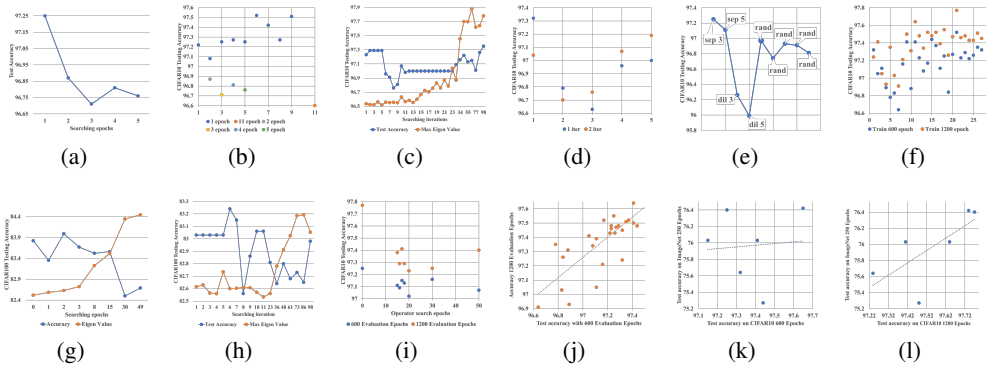


Figure 3: Ablation study. (a): CIFAR10: Accuracy - Search epochs (in the same Run); (b): CIFAR10: Accuracy - Search epochs (multiple runs); (c): CIFAR10: Accuracy - Search iterations; (d): Accuracy on CIFAR10: 1 vs 2 search iterations; (e): CIFAR10: Accuracy - Operator replacing skip-connections; (f): CIFAR10: Accuracy - Evaluation epochs; (g): Epoch-wise on CIFAR100: Accuracy - Max eigenvalue; (h): Iteration-wise on CIFAR100: Accuracy - Max eigenvalue; (i): CIFAR10: Operator search's epoch-wise impact on testing accuracy (0 epoch means only searching for the topology); (j): CIFAR10: Evaluation epochs' impacts on the testing accuracy (same architecture); (k):Testing accuracy: CIFAR10 600 evaluation epochs - ImageNet 250 evaluation epochs (same architecture); (l): Testing accuracy: CIFAR10 1200 evaluation epochs - ImageNet 250 evaluation epochs (same architecture)

When we do not search for the operators, after the topology search, we assign all the remaining edges a fixed operator. Thus, which operator we should choose becomes a critical question. We show in Figure 3(e) that 3×3 *separable convolution* can indeed outperform all other operators in terms of accuracy, as discussed in 4.1, and randomly assigning each edge an operator among all the candidate operators, except for *None*, can lead to comparable testing accuracy. If we want to extend FTSO to other tasks, first searching for the topology with the simplest operator, and then replacing it with the most promising one should be the basic principle.

As shown in Figure 3(f), we find that under a different number of evaluation epochs, both the absolute value and the relative ranking of the same architecture's testing accuracy may vary; i.e., some architectures performing well within 600 epochs perform poorly after 1200 epochs. However, in general, they still obey a positive correlation, with a Pearson correlation coefficient of 0.77, as shown in Figure 3(j). In terms of the generalization ability from CIFAR10 to ImageNet, Figure 3(l) reveals that the architectures performing well after long-term evaluation on CIFAR10 can usually generalize better on ImageNet, with the correlation coefficient of 0.7; yet, as shown in Figure 3(k), there is no guarantee that those working well on CIFAR10 within limited evaluation epochs can also dominate on ImageNet because this only proves that they can quickly converge, not that they can converge to a global optimum.

Another important fact, as shown in Figure 3(i), is that, coinciding with our analysis in 4.1, for an architecture already performing well after the topology search, an additional operator search may bring no benefit. This is because the operator search requires a large supernet, which can easily lead to over-fitting.

In Figure 3(g) and 3(h), we show that one epoch is not only an optimal choice on CIFAR10, but also enough for the topology-only search on CIFAR100. In addition, as the search epochs and iterations increase, the max eigenvalue of the loss's Hessian matrix on the validation set increases and the

Table 3: Different configurations' impacts to FTSO

FTSO's Configuration	**CIFAR10 Err.(%)**		**ImageNet Err.(%)**		**Search Cost (GPU-days)**	
	600	**1200**	**Top1**	**Top5**	**CIFAR**	**ImageNet**
CIF. Topo(skip,1it.)	2.68	2.54	24.36	7.27	7.87×10^{-6}	-
CIF. Topo(skip,1ep.)	2.48	2.23	23.60	7.01	2×10^{-4}	-
CIF. Topo(skip,50ep.)	2.77	2.52	-	-	0.01	-
CIF. Topo(skip,1ep.)+Op(18ep.)	2.85	2.59	23.97	7.20	0.01	-
CIF. Topo(skip,50ep.)+Op(50ep.)	2.59	2.36	23.97	7.12	0.05	-
CIF. Topo(m.p.,50ep.)+Op(50ep.)	2.83	2.48	-	-	0.05	-
CIF. Topo(sep3,50ep.)	2.63	2.48	-	-	0.02	-
CIF. Topo(sep3,50ep.)+Op(50ep.)	2.56	2.52	24.73	7.60	0.06	-
CIF. Topo(3op,50ep.)+Op(50ep.)	2.59	2.50	-	-	0.05	-
CIF. Topo(4op,50ep.)+Op(50ep.)	2.68	2.59	-	-	0.07	-
Part ImageNet Topo(skip,1it.)	-	-	24.03	7.07	-	0.0002
Part ImageNet Topo(skip,1ep.)	-	-	23.94	7.05	-	0.0017
Part ImageNet Topo(skip,6ep.)	-	-	24.59	7.38	-	0.009
Full ImageNet Topo(skip,1ep.)	2.35	2.26	23.58	6.80	-	0.01

* '3op' means *max pool 3x3*, *skip connect* and *none*. '4op' means: *sep conv 3x3*, *max pool 3x3*, *skip connect* and *none*. 'm.p.' means: *max pool 3x3*. 'sep3' means: *sep conv 3x3*

testing accuracy is generally decreasing because the model's generalization ability is dropping. This phenomenon is particularly obvious epoch-wise because, after just a few iterations, the model can already reach a comparable accuracy on the training set. Then the model's performance on the testing set starts to relate with its generalization ability.

5 Generalization to other tasks and search spaces

Although in this paper, we establish FTSO in a differentiable search space, in fact, the first topology second operator strategy is not limited within any specific search space or tasks. No matter the search space is discrete or continuous, we first shrink the candidate operator set, and only retain the simplest operator. After this, the size of the whole search space is reduced in magnitude. Then, we search for the best topology with whatever search algorithm. In this way, a promising topology can be found. Now we have two choices. The first solution is to form a new search space with both the found topology and all the candidate operators, then utilize whatever search algorithm to find the best operators. The second solution is to assign all the edges a fixed operator. Here the principle is to pick the operators performing the best after fully converging. However, when the searching set is not informative enough, picking an operator with slightly fewer parameters should be a better choice. As for the selection between these two solutions. If the search algorithm is hard to over-fit, choosing the first solution may be better. Otherwise, the second solution might be more promising.

6 Conclusion

In this paper, we propose an ultra computationally efficient neural architecture search method named FTSO, which reduces NAS's search time cost from days to less than 0.68 seconds, while achieving 1.5% and 0.27% accuracy improvement on ImageNet and CIFAR10 respectively. Our key idea is to divide the search procedure into two sub-phases. In the first phase, we only search for the network's topology with simple operators. Then, in the second phase, we fix the topology and only consider which operators we should choose.

Our strategy is concise in both theory and implementation, and our promising experimental results show that current NAS methods contain too much redundancy, which heavily impacts the efficiency and becomes a barrier to higher accuracy. What is more, our method is not bound by differentiable search spaces, so we believe that our key idea can also cooperate well with existing NAS approaches.

REFERENCES

Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=S1c2cvqee.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. 2020.

Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HylVB3AqYm.

X. Chen, L. Xie, J. Wu, and Q. Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1294–1303, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

Adam Gaier and David Ha. Weight agnostic neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 5364–5378. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/8777-weight-agnostic-neural-networks.pdf.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1eYHoC5FX.

Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4095–4104, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/pham18a.html.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 4780–4789. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33014780. URL https://doi.org/10.1609/aaai.v33i01.33014780.

Yao Shu, Wei Wang, and Shaofeng Cai. Understanding architectures learnt by cell-based neural architecture search. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJxH22EKPS.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.1556.

C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015.

Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. SNAS: stochastic neural architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rylqooRqK7.

Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJlS634tPr.

Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1gDNyrKDS.

Tunhou Zhang, Hsin-Pai Cheng, Zhenwen Li, Feng Yan, Chengyu Huang, Hai Li, and Yiran Chen. Autoshrink: A topology-aware nas for discovering efficient neural architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2020)*, 2019.

Hongpeng Zhou, Minghao Yang, Jun Wang, and Wei Pan. Bayesnas: A bayesian approach for neural architecture search. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7603–7613. PMLR, 2019. URL http://proceedings.mlr.press/v97/zhou19e.html.

B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8697–8710, 2018.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=r1Ue8Hcxg.