Article

FTSO: Effective NAS via First Topology Second Operator

Likang Wang [*] and Lei Chen

Posted Date: 29 March 2023

doi: 10.20944/preprints202303.0345.v2

Keywords: neural architecture search; machine learning; computer vision

Article

FTSO: Effective NAS via First Topology Second Operator

Likang Wang [1,*] and Lei Chen [1,2]

[1] Department of Computer Science and Technology, The Hong Kong University of Science and Technology; leichen@cse.ust.hk
[2] Data Science and Analytics Thrust, The Hong Kong University of Science and Technology (Guangzhou)
[*] Correspondence: lwangcg@connect.ust.hk

Abstract: Existing one-shot neural architecture search (NAS) methods have to conduct a search over a giant super-net, which leads to the huge computational cost. To reduce such cost, in this paper, we propose a method, called FTSO, to divide the whole architecture search into two sub-steps. Specifically, in the first step, we only search for the topology, and in the second step, we search for the operators. FTSO not only reduces NAS's search time from days to 0.68 seconds, but also significantly improves the found architecture's accuracy. Our extensive experiments on ImageNet show that within 18 seconds, FTSO can achieve a 76.4% testing accuracy, 1.5% higher than the SOTA, PC-DARTS. In addition, FTSO can reach a 97.77% testing accuracy, 0.27% higher than the SOTA, with nearly 100% (99.8%) search time saved, when searching on CIFAR10.

Keywords: machine learning; neural architecture search; nas; darts; pc-darts; computer vision; image classification

---

1. Introduction

Since the great success of the AlexNet [7] in image classification, most modern machine learning models [16–18] have been developed 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 [5,14,15] 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 due to the expensive time and resource overheads. To ease this problem, a new technique called neural architecture search (NAS) was introduced. It allows computers to search for better network architectures automatically instead of relying on human experts.

Early-proposed reinforcement learning-based NAS methods [1,23,24] typically have an RNN-based controller to sample candidate network architectures from the search space. Although these algorithms can provide promising accuracy, their computation cost is usually unaffordable, for instance, 1800 GPU-days are required for NASNet to find an image classification network on CIFAR10.

To ease the search efficiency problem, one-shot approaches [2,9,10] with parameter sharing have been proposed. These methods first create a huge directed acyclic graph (DAG) super-net, containing the whole search space. Then, the kernel weights are shared among all the sampled architectures via the super-net. This strategy makes it possible to measure the candidate architecture's performance without repeatedly retraining it from scratch. However, these algorithms suffer from the super-nets' computational overheads. This problem is particularly severe for differentiable models [9,20].

Limited by current NAS algorithms' inefficiency, it is rather challenging to find satisfying network architectures on large-scale datasets and complex tasks. For instance, current speed-oriented NAS approaches generally require days to accomplish one search trial on ImageNet, for example, 8.3 GPU-days for ProxylessNAS [2] and 3.8 GPU-days for PC-DARTS [20]. Therefore, we argue that it is essential to propose a new well-defined search space, which is not only expressive enough to cover the most powerful architectures, but also compact enough to filter out the poor architectures.

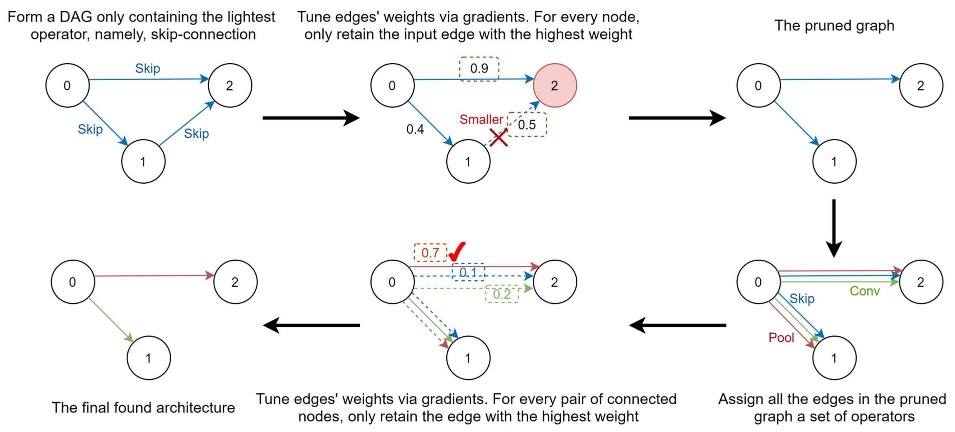
Form a DAG only containing the lightest operator, namely, skip-connection
Tune edges' weights via gradients. For every node, only retain the input edge with the highest weight
The pruned graph
Skip
0.9
Smaller
0.5
0.4
Assign all the edges in the pruned graph a set of operators
Skip
Conv
Pool
0.7
0.1
0.2
Tune edges' weights via gradients. For every pair of connected nodes, only retain the edge with the highest weight
The final found architecture

Figure 1. The main structure of FTSO.

We are motivated by Shu et al. [13], who demonstrate that randomly replacing operators in a found architecture does not harm the accuracy much. As such, we believe that there would be no reduction in the test accuracy if we omit the influence of operators and cluster architectures according to the topology. 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).

In this paper, we first mathematically prove that FTSO reduces the number of network parameters by 10^8, decreases the FLOPs per iteration by 10^5 and lowers the operator's complexity in magnitude. We then empirically reveal that FTSO shortens the required search period from 50 epochs to one iteration. Besides the great improvement in efficiency, FTSO also significantly promotes the effectiveness by easing the over-fitting phenomenon and the Matthew effect [13]. To be specific, each architecture in DARTS has only one iteration to tune its kernel weights, and within one iteration, only the operators with few parameters may converge. The result is that the simpler operators outperform the more powerful ones in the super-net, then larger gradients to enlarge their advantages are achieved. In this way, the found architectures tend to only contain the simplest operators and perform poorly on both the training and testing sets. Such a phenomenon is called the Matthew effect.

Our extensive experiments show that FTSO can accomplish the whole architecture search in 0.68 seconds. On ImageNet, FTSO achieves 76.4% testing accuracy, 1.5% higher than the SOTA, within a mere 18 seconds. More importantly, when we only search for one iteration, FTSO consumes less than 0.68 seconds, while reaching 75.64% testing accuracy, 0.74% higher than the SOTA. Moreover, if we allow FTSO to search for 19 minutes, 76.42% Top1 and 93.2% Top5 testing accuracy can be achieved. In addition, FTSO can reach 97.77% testing accuracy, 0.27% higher than the SOTA, with nearly 100% (99.8%) search time saved, when searching on CIFAR10. Although in this paper we have implemented FTSO within a continuous search space, we illustrate in Section 5 that FTSO can be seamlessly transferred to other NAS algorithms.

2. Related Work

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 [23,24] generally suffer from high computational cost and low-efficiency sampling. Instead of sampling a discrete architecture and then evaluating it, DARTS [9] treats the whole search space as a continuous super-net. 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 super-net. After the super-net 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 super-net, which inevitably leads to heavy computational overheads and over-fitting.

To relieve the computational overhead of DARTS, DARTS-ES [21] reduces the number of searching epochs via early stopping, according to the Hessian matrix's max eigenvalue. PC-DARTS [20] decreases the FLOPs per iteration by only calculating a proportion of the input channels and keeping 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.

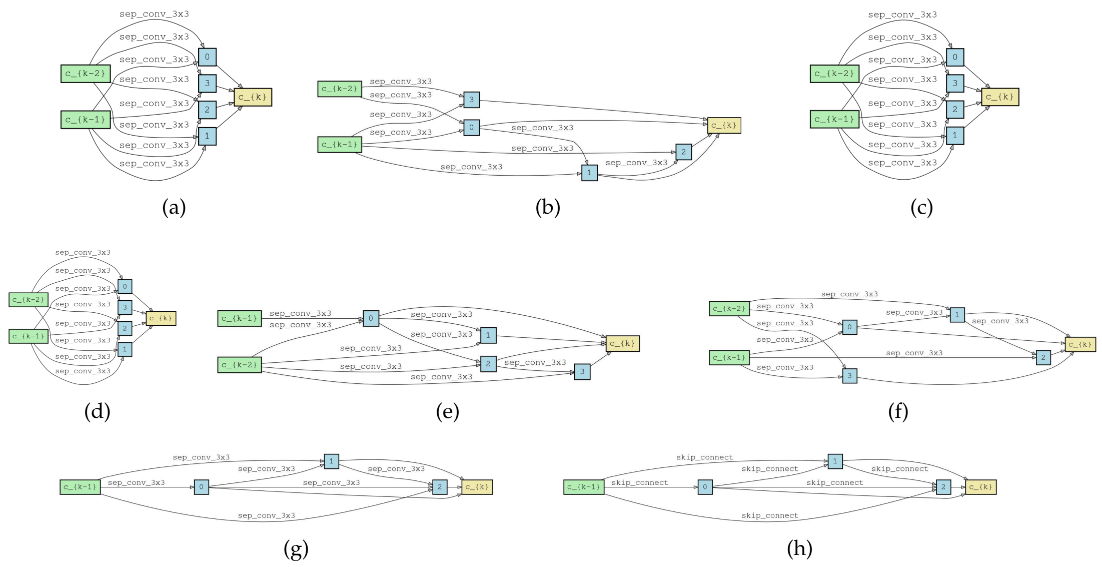

Figure 2. FTSO's found architectures. (a) and (b): Normal and reduction cells found on CIFAR10 after one epoch's search; (c) and (d): Normal and reduction cells found on the entire ImageNet after one epoch's search; (e) and (f): Normal and reduction cells found on CIFAR10, where we adopt the operator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

3. FTSO: Effective NAS via First Topology Second Operator

Existing NAS approaches generally suffer from a huge computational overhead and an unsatisfying testing accuracy led 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 of great demand to investigate the correlations among different architectures and to shrink the search space according to the prior knowledge. We notice that there is an important observation in Shu et al. [13] that randomly substituting the operators in a found architecture does not observably influence the testing accuracy. Therefore, it would be great inspiration if we could cluster the architectures according to their connection topologies. To be specific, suppose we find an architecture only containing the simplest operators achieves high accuracy on the testing set, if we replace all the connections in this architecture with powerful operators, the converted architecture can perform well on the testing set with high confidence.

In this paper, we first propose to find the most effective network topology with simple operators. We 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 shrunk in magnitude. We name this new NAS algorithm Effective NAS via First Topology Second Operator (FTSO).

We summarize the symbols used in this section in Table 2. As shown in Figure 1, we inherit the differentiable framework of PC-DARTS, and divide the architecture search into two phases. We name the two phases topology search and operator search, and illustrate how they work in Algorithms 1 and 2, respectively. In the first phase, we form a super-net only containing the simplest operator, *skip connection*. Since the *skip connection* operator contains no kernel weights, we only need to optimize the architecture parameters $\beta_{i,j}$. In fact, as shown in Table 1, the *max pooling* operator also delivers satisfying results for the topology search. There are two reasons we use *skip connection*.

Table 1. Different configurations' impacts to FTSO.

FTSO's Configuration	**CIFAR10 Error (%)**		**ImageNet Error (%)**		**Search Cost (GPU-days)**	
	600 Epoch	**1200 Epoch**	**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*. 'CIF.' means CIFAR10; 'Topo(skip,1it)' means to search for topology with only skip connections for 1 iteration; '1ep' means 1 epoch; 'Part ImageNet' means to search on part of ImageNet.

Table 2. Symbol table.

Symbol	**Meaning**
$\alpha_{i,j}^{o}$	Weight for operator o from node i to j
$\beta_{i,j}$	Node i's importance to the node j
$f_{i,j}$	Linear combination of all the candidate operators o with weight $\alpha_{i,j}^{o}$

Algorithm 1 Topology search

```
Require: a set of nodes: $n_k$
Ensure: the pruned architecture: $A_p$
  1. Create an directed edge $e_{i,j}$ with weight $\beta_{i,j}$ between each pair of nodes $n_i$ and $n_j$ $(i < j)$
  2. Assign each edge $e_{i,j}$ a *skip connection* operator $o_{i,j}$ with kernel weights $w_{i,j}$
  while still in the first epoch do
    1. Forward-propagate following $n_j = \sum_{i<j} o(n_i)\beta_{i,j}$
    2. Update architecture $\beta$ by descending $\nabla_\beta \mathcal{L}_{val}(w, \beta)$
    3. Update weights $w$ by descending $\nabla_w \mathcal{L}_{train}(w, \beta)$
  end while
  for each node $n_j \in A_p$ do
    $T_j \leftarrow$ the second largest $\beta_{i,j}$
    for each node $n_i$ do
      if $\beta_{i,j} < T_j$ then
        Prune edge $e_{i,j}$
      end if
    end for
  end for
  Derive the pruned architecture $A_p$.
```

Algorithm 2 Operator search

```
Require: the pruned architecture produced by the topology search: $A_p$
Ensure: the found architecture: $A_f$
  if replace with convolutions then
    Replace all the retained operators $o_{i,j}$ in $A_p$ with convolutions
  else
    Each node $n_j \leftarrow \sum_{i<j} \sum_{o \in \mathcal{O}} \frac{\exp \alpha^o_{i,j}}{\sum_{o' \in \mathcal{O}} \exp \alpha^{o'}_{i,j}} o(n_i)$
    while not converged do
      Update architecture $\alpha$ by descending $\nabla_\alpha \mathcal{L}_{val}(w, \alpha)$
      Update weights $w$ by descending $\nabla_w \mathcal{L}_{train}(w, \alpha)$
    end while
  end if
  for each edge $e_{i,j} \in A_p$ do
    Assign edge $e_{i,j}$ the operator $o' \in \mathcal{O}$ with the highest $\alpha^{o'}_{i,j}$
  end for
  Derive the found architecture $A_f \leftarrow A_p$.
```

The first is that the *skip connection* operator not only requires zero parameters, but also demands the minimum computational cost. The second reason is that *max pooling* may lead to the loss of useful information if the network is deep. Furthermore, as the only difference between our topology search and the vanilla DARTS is the number of candidate operators, the pruned architecture's connectivity can be guaranteed.

Similar to DARTS and PC-DARTS, 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 pruned topology with two strategies. The first strategy is similar to the vanilla DARTS. It replaces each operator in the pruned topology with a mix-operator $f_{i,j}$. After that, we optimize the architecture parameters, $\alpha^o_{i,j}$, $\beta_{i,j}$ and the kernel weights $\omega^o_{i,j}$ alternatively. After the super-net converges, we only retain one operator o^* with the highest $\alpha^o_{i,j}$ for every two connected nodes i and j. The second strategy directly replaces all the operators in the pruned topology with one single operator owning the highest model capacity, e.g., a convolution operator.

In this paper, we take the second strategy as the default configuration because it is not only much more efficient, but also avoids the over-fitting phenomenon and the Matthew effect in DARTS. To be specific, in DARTS-like methods, suppose the found network perfectly fits the data, then the super-net must severely over-fit, since the super-net is much larger than the found network. As we know, an over-fitting model can hardly generalize well. Thus, the generated sub-graph is not likely to be the best architecture. While in the second strategy, since no super-net is adopted and all the simple operators in the sub-graph are replaced with powerful operators, the final architecture's model capacity gets promoted. Additional empirical comparisons between these two strategies can be found in Section 4.4.

In DARTS, the network topology and operators are jointly searched, which makes both the size and the computational cost of the super-net extremely high. We use n to denote the number of nodes and p to denote the number of candidate operators. Since we have two input nodes, one output node and $n-3$ intermediate nodes, the super-net contains a total of $\frac{1}{2}(n^2-3n)$ edges. At the same time, every edge keeps p operators, thus, the total number of operators in DARTS is $\frac{1}{2}(n^2-3n)p$. 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. This is because in the topology search, every edge contains only one operator; and in the topology search, every intermediate node only connects to two predecessors. Since n is usually close to p, FTSO reduces the number of operations from $O(n^3)$ to $O(n^2)$.

Theorem 1. *The total number of FLOPs and parameters of DARTS are* $\frac{1}{2}pn(n-3)H_{out}W_{out}C_{out}(k^2C_{in}+1)$ *and* $\frac{1}{2}n(n-3)p(k^2C_{in}+1)C_{out}$ *respectively.*

Proof. Each 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. For simplicity, assume all the candidate operators are convolutions. Since DARTS has $\frac{1}{2}pn(n-3)$ edges, it needs to compute $\frac{1}{2}pn(n-3)$ convolutions and $\frac{1}{2}pn(n-3)$ tensor summations. Owing to each tensor summation consuming $H_{in}W_{in}C_{in}$ FLOPs, DARTS requires a total of $\frac{1}{2}pk^2n(n-3)C_{in}H_{out}W_{out}C_{out}$ convolution FLOPs and $\frac{1}{2}pn(n-3)H_{out}W_{out}C_{out}$ summation FLOPs. Thus, the overall FLOPs are parameters are $\frac{1}{2}pn(n-3)H_{out}W_{out}C_{out}(k^2C_{in}+1)$ and $\frac{1}{2}n(n-3)p(k^2C_{in}+1)C_{out}$, respectively. □

Theorem 2. *The total number of FLOPs and parameters of FTSO are* $\frac{1}{2}n(n-3)H_{in}W_{in}C_{in}$ *and* $\frac{1}{2}n(n-3)$ *respectively.*

Proof. Each *skip connection* operator needs 0 parameters and 0 FLOPs. If we first search for the topology and then directly substitute the operators, only $\frac{1}{2}n(n-3)$ tensor summations need to be calculated, since FTSO has $\frac{1}{2}n(n-3)$ operators. □

In addition to the reduction in the number of operations, FTSO also dramatically decreases the internal cost of the operations. This is because during the topology search all the powerful operators are replaced by the simple operators. We summarize the main properties of DARTS and FTSO in Theorems 1 and 2. As a typical configuration, let $k=5$, $C_{in}=C_{out}=512$, $n=7$, $p=8$. Then, our algorithm requires only $\frac{1}{p(k^2C_{in}+1)C_{out}}=1.9\times10^{-8}$ times the parameters and $\frac{1}{p(k^2C_{in}+1)}=9.8\times10^{-6}$ times the forward-propagation FLOPs per iteration compared to those of DARTS.

FTSO's huge reduction on the parameter numbers provides us a large number of benefits. As mentioned above, it allows the algorithm to converge in only a few iterations and prevents over-fitting. This is because when extracting the discrete sub-graph from the super-net, many architecture parameters are set to 0. The introduced disturbance impacts more on the over-fitting super-nets since they prefer sharper local minimums. Furthermore, Since FTSO only contains one operator with 0 parameters, the Matthew effect is eliminated.

4. Experiments

Our search algorithm is evaluated on the three most widely-used datasets in NAS papers, namely, CIFAR10 [6], ImageNet [12] and NATS-Bench [4]. Following DARTS, our search space contains a total of eight candidate operators: 3×3 and 5×5 *separable convolutions*, 3×3 and 5×5 *dilated*

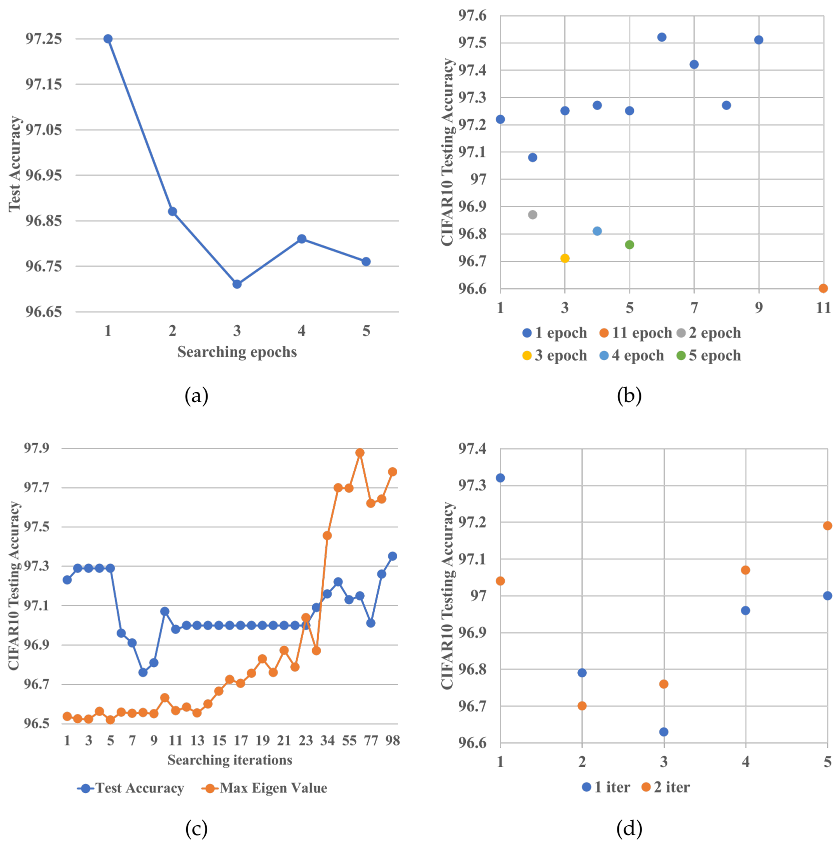

Figure 3. Ablation study Part 1. (a): CIFAR10: Accuracy - Search epochs (in the same run); (b): CIFAR10: Accuracy - Search epochs (multiple runs); (c): CIFAR10: Accuracy - Search iterations; (d): Accuracy on CIFAR10: 1 vs 2 search iterations (multiple runs).

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. As mentioned in Section 3, we have two strategies to determine the operators, including the one based on gradients and the one directly replacing operators. In Sections 4.1 and 4.2, we focus more on the second configuration. In Section 4.4, we have a comprehensive comparison on these two strategies. All detailed configurations are shown in the Appendices. In addition, most of our experiments only search for one epoch or one iteration because of the benefits of FTSO's huge reduction on the parameter numbers. For more experimental support, please refer to Section 4.4. 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 super-nets contain a large amount of parameters, which require a long period to tune.

4.1. Results on CIFAR10

We compare FTSO to existing state-of-the-art NAS methods in Table 3. In the experiment, we only search for the topology with skip-connections and then replace them all with 3×3 *separable convolutions*. The reason that 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 becomes 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.

Table 3. Comparison with existing state-of-the-art image classification architectures.

Architecture	CIFAR Err. (%)		ImageNet Err. (%)		Search Cost (GPU-days)	
	600 Ep.	1200 Ep.	Top1	Top5	CIFAR	ImageNet
NASNet-A†[24]	2.65	-	26.0	8.4	1800	-
AmoebaNet†[11]	2.55	-	24.3	7.6	3150	-
PNAS[8]	3.41	-	25.8	8.1	225	-
ENAS†[10]	2.89	-	-	-	0.5	-
DARTS (2nd)†[9]	2.76	-	26.7	8.7	1	4.0
SNAS†[19]	2.85	-	27.3	9.2	1.5	
ProxylessNAS†*[2]	2.08	-	24.9	7.5	4.0	8.3
P-DARTS†[3]	2.50	-	24.4	7.4	0.3	-
BayesNAS†[22]	2.81	-	26.5	8.9	0.2	-
PC-DARTS(CIFAR10)†[20]	2.57	2.50	25.1	7.8	0.1	-
PC-DARTS(ImageNet)†*[20]	-	-	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)†*	**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.

On the other hand, suppose both the image's resolution and the dataset's scale are big enough and the evaluation period is adequate, the 5×5 *separable convolution* might be a better choice. The found architecture after one epoch's search is shown in Figure 2. Due to FTSO containing only a few trainable parameters, it can even achieve comparable accuracy to PC-DARTS with only a one-time gradient update. Under this configuration, a mere 0.68 seconds are required and 99.993% of the search time is saved. In addition, as shown in Table 1, when the topology is searched with powerful operators for a long period, an 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.

4.2. Results on ImageNet

On ImageNet, we use similar configurations to those on CIFAR10. When searching, we have two configurations. The detailed configurations are shown in the Appendices. Our experiments in Table 3 show that FTSO is significantly superior to existing methods in both efficiency and effectiveness. 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' architectures shown in Figure 2(e) and 2(f), which were 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 this is because the whole model is stacked with many cells. If the depth of each cell is too high, it leads to a very deep neural network. At that time, because all the operators in our found architecture are convolutions, we cannot use skip connections to facilitate gradients' propagation in ResNet's manner. In this way, both the vanishing and explosion of gradients may prevent the deeper models from higher performance.

4.3. Results on NATS-Bench

In the search space of NATS-Bench, there is one input node, three intermediate nodes and one output node, and each intermediate node connects to all its predecessors. Here we implement FTSO based on DARTS instead of PC-DARTS, and we compare FTSO's performance to other NAS algorithms in Table 4. It is shown that FTSO dominates DARTS in all configurations. Coinciding with our analysis in Section 3, the architectures found by DARTS tend to only contain simple operators, thus cannot achieve satisfying accuracy. For example, when searching on CIFAR10, the architecture found by DARTS is full of *skip connections* as shown in Figure 2(h). By comparison, as shown in Figure 2(g), the architecture found by FTSO is much more powerful.

Table 4. Comparison with existing state-of-the-art image classification architectures (NATS-Bench).

Architecture	**Search on CIFAR10**			**Search on CIFAR100**			**Search on ImageNet**		
	CF10*	**CF100**	**ImageNet**	**CF10**	**CF100**	**ImageNet**	**CF10**	**CF100**	**ImageNet**
DARTS (1st)	54.30	15.61	16.32†	86.57	58.13	28.50	89.58	63.89	33.77
DARTS (2nd)	86.88	58.61	28.91	91.96	67.27	39.47	84.64	55.15	26.06
FTSO	93.98	70.22	45.57	93.98	70.22	45.57	93.98	70.22	45.57

† This means that within NATS-Bench's search space, when we use the 1st order DARTS to search for architectures on the CIFAR10 dataset, the found architecture can achieve 16.32% testing accuracy on ImageNet. * CF10 means testing accuracy (%) on CIFAR10; CF100 means testing accuracy (%) on CIFAR100

4.4. Ablation Study

In terms of a 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. We find that the found architecture's performance generally first drops and then increases in the first epoch, and then always decreases after the first epoch. In Figure 3(c) we show that although one iteration cannot surpass one epoch, it is better than a few iterations. This is because when we only search for one iteration, the model does not over-fit the data, thus the model generalizes well. When we only searched for a few iterations, the number of different images seen by the model is not big enough. However, since the super-net only contains skip connections, such a number of gradient updates is enough for the architecture parameters to become over-fitted. This is the reason that a few iterations perform worse than one iteration. After we have searched for one whole epoch, the super-net has seen enormous different images, which helps it to generalize better on the testing set. This is the reason one epoch performs the best. In terms of whether we should search for one iteration or two, in Figure 3(d), we show that both choices work well. 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. Figure 4(a) show that a 3×3 *separable convolution* can indeed outperform all other operators in terms of accuracy.

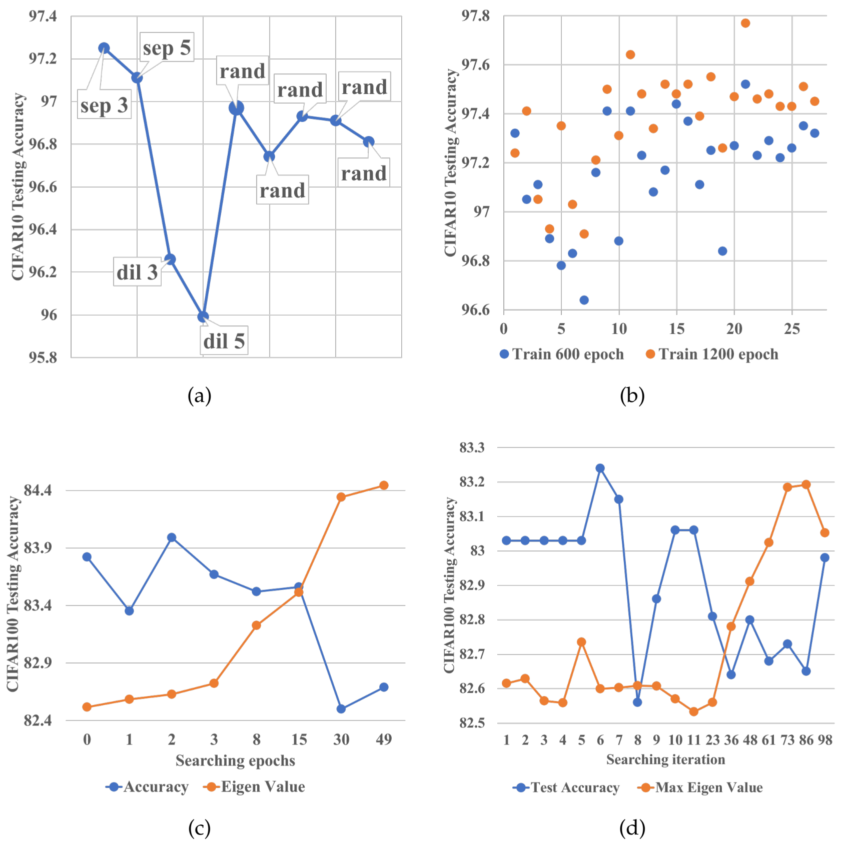

Figure 4. Ablation study Part 2. (a): CIFAR10: Accuracy - Operator replacing skip-connections; (b): CIFAR10: Accuracy - Training epochs of the found architecture; (c): Epoch-wise on CIFAR100: Accuracy - Max eigenvalue $\nabla^2_{Arch}\mathcal{L}_{val}$; (d): Iteration-wise on CIFAR100: Accuracy - Max eigenvalue $\nabla^2_{Arch}\mathcal{L}_{val}$.

As shown in Figure 4(b), 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 which perform 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 5(b). In terms of the generalization ability from CIFAR10 to ImageNet, Figure 5(d) reveals that the architectures which perform well after long-term evaluation on CIFAR10 can usually generalize better on ImageNet, with a correlation coefficient of 0.7; yet, as shown in Figure 5(c), there is no guarantee that those working well on CIFAR10 within limited evaluation epochs can also dominate on ImageNet. This is because it only proves that they can converge quickly, not that they can converge to a global optimum.

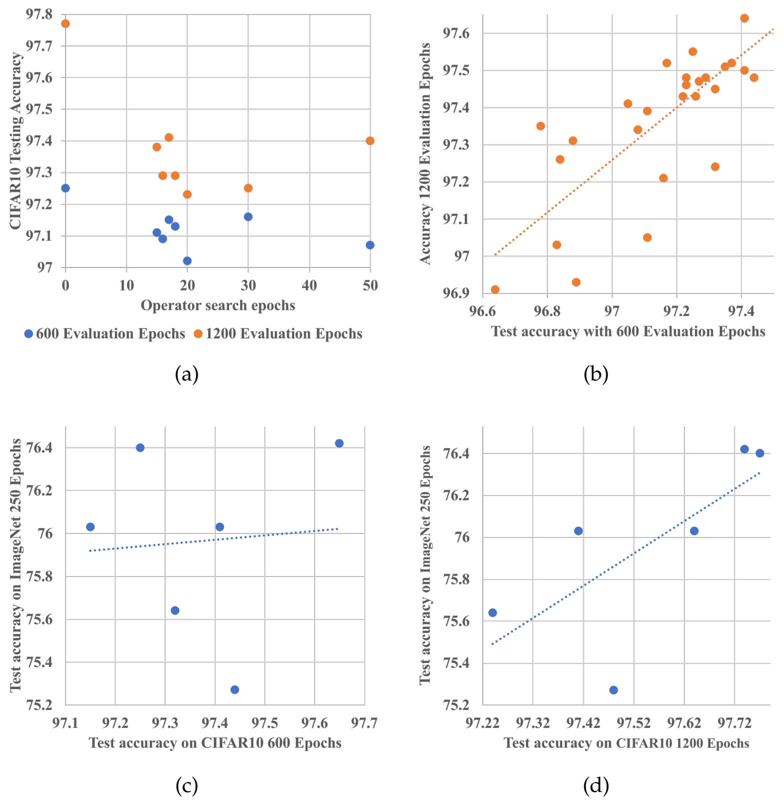

Figure 5. Ablation study Part 3. (a): CIFAR10: Operator search's epoch-wise impact on testing accuracy (0 epoch means only searching for the topology); (b): CIFAR10: The found architecture's training epochs' impacts on the testing accuracy (same architecture); (c):Testing accuracy: CIFAR10 600 evaluation epochs - ImageNet 250 evaluation epochs (evaluating after training the same architecture on different datasets); (d): Testing accuracy: CIFAR10 1200 evaluation epochs - ImageNet 250 evaluation epochs (evaluating after training the same architecture on different datasets)

In Figures 4(c) and 4(d), 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. At the same time, the testing accuracy generally decreases 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

As shown in Section 4, FTSO works well under different search spaces and node numbers. Theoretically, FTSO's advantages to DARTS can be enlarged while the search space and node number increase. The reason is that FTSO reduces the computational cost from $O(n^3)$ to $O(n^2)$ and avoids over-fitting. Based on such considerations, as the future work, we plan to apply FTSO in high-level tasks, for example, instance segmentation and multi-view stereo. Although in this paper, we establish FTSO within differentiable search spaces, in fact, the first topology second operator strategy is not limited to any specific search space or tasks. Whether or not the search space is discrete or continuous, or the search algorithm is gradient-based or reinforcement learning-based, we first shrink the candidate operator set, and only retain the simplest operator, in language modeling which might be a *skip connection* or a *pooling layer*. After this, the size of the whole search space is reduced in magnitude. Then, we search for the best topology with any available search algorithm. In this way, a promising topology can be found. Then, we can either directly assign each edge a powerful operator, in language modeling which might be a *LSTM unit* or an *attention layer*, or use gradients to search for operators. Generally, the directly replacing strategy leads to higher accuracy, and the gradient-based strategy reduces the model complexity.

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, as mentioned in Section 5, our method is not bound by differentiable search spaces as it can also cooperate well with existing NAS approaches.

References

1. Baker, B., Gupta, O., Naik, N., and Raskar, R. 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.
2. Cai, H., Zhu, L., and Han, S. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2019.
3. Chen, X., Xie, L., Wu, J., and Tian, Q. 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.
4. Dong, X., Liu, L., Musial, K., and Gabrys, B. NATS-Bench: Benchmarking nas algorithms for architecture topology and size. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021. doi: 10.1109/TPAMI.2021.3054824. doi:10.1109/TPAMI.2021.3054824.
5. He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.
6. Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.
7. Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012. URL http://t.cn/RtdxBCE.
8. Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., and Murphy, K. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.
9. Liu, H., Simonyan, K., and Yang, Y. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019.
10. Pham, H., Guan, M., Zoph, B., Le, Q., and Dean, J. Efficient neural architecture search via parameters sharing. In Dy, J. and Krause, A. (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.
11. Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. 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.

12. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.
13. Shu, Y., Wang, W., and Cai, S. Understanding architectures learnt by cell-based neural architecture search. In *International Conference on Learning Representations*, 2020.
14. Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Bengio, Y. and LeCun, Y. (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.
15. Szegedy, C., Wei Liu, Yangqing Jia, Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015.
16. Wang, L. and Chen, L. Dionysus: Recovering scene structures by dividing into semantic pieces. *ResearchGate*, 2023.
17. Wang, L., Gong, Y., Ma, X., Wang, Q., Zhou, K., and Chen, L. Is-mvsnet:importance sampling-based mvsnet. In Avidan, S., Brostow, G., Cissé, M., Farinella, G. M., and Hassner, T. (eds.), *Computer Vision – ECCV 2022*, pp. 668–683, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-19824-3.
18. Wang, L., Gong, Y., Wang, Q., Zhou, K., and Chen, L. Flora: dual-frequency loss-compensated real-time monocular 3d video reconstruction. *Preprints*, 2023.
19. Xie, S., Zheng, H., Liu, C., and Lin, L. SNAS: stochastic neural architecture search. In *International Conference on Learning Representations*, 2019.
20. Xu, Y., Xie, L., Zhang, X., Chen, X., Qi, G.-J., Tian, Q., and Xiong, H. Pc-darts: Partial channel connections for memory-efficient architecture search. In *International Conference on Learning Representations*, 2020.
21. Zela, A., Elsken, T., Saikia, T., Marrakchi, Y., Brox, T., and Hutter, F. Understanding and robustifying differentiable architecture search. In *International Conference on Learning Representations*, 2020.
22. Zhou, H., Yang, M., Wang, J., and Pan, W. Bayesnas: A bayesian approach for neural architecture search. In Chaudhuri, K. and Salakhutdinov, R. (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.
23. Zoph, B. and Le, Q. V. 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.
24. Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8697–8710, 2018.