Block-wisely Supervised Neural Architecture Search with Knowledge Distillation

Changlin Li[2]*, Jiefeng Peng[1]*, Liuchun Yuan[1,3], Guangrun Wang[1,3]†, Xiaodan Liang[1,3], Liang Lin[1,3], Xiaojun Chang[2]

[1]DarkMatter AI Research [2]Monash University [3]Sun Yat-sen University

{changlin.li,xiaojun.chang}@monash.edu, {jiefengpeng,ylc0003,xdliang328}@gmail.com, wanggrun@mail2.sysu.edu.cn, linliang@ieee.org

Abstract

Neural Architecture Search (NAS), aiming at automatically designing network architectures by machines, is expected to bring about a new revolution in machine learning. Despite these high expectation, the effectiveness and efficiency of existing NAS solutions are unclear, with some recent works going so far as to suggest that many existing NAS solutions are no better than random architecture selection. The ineffectiveness of NAS solutions may be attributed to inaccurate architecture evaluation. Specifically, to speed up NAS, recent works have proposed under-training different candidate architectures in a large search space concurrently by using shared network parameters; however, this has resulted in incorrect architecture ratings and furthered the ineffectiveness of NAS.

In this work, we propose to modularize the large search space of NAS into blocks to ensure that the potential candidate architectures are fully trained; this reduces the representation shift caused by the shared parameters and leads to the correct rating of the candidates. Thanks to the block-wise search, we can also evaluate all of the candidate architectures within each block. Moreover, we find that the knowledge of a network model lies not only in the network parameters but also in the network architecture. Therefore, we propose to distill the neural architecture (DNA) knowledge from a teacher model to supervise our block-wise architecture search, which significantly improves the effectiveness of NAS. Remarkably, the performance of our searched architectures has exceeded the teacher model, demonstrating the practicability of our method. Finally, our method achieves a state-of-the-art 78.4% top-1 accuracy on ImageNet in a mobile setting. All of our searched models along with the evaluation code are available at https://github.com/changlin31/DNA.

*Changlin Li and Jiefeng Peng contribute equally and share first-authorship. This work was done when Changlin Li worked as an intern at DarkMatter AI.

†Corresponding Author is Guangrun Wang.

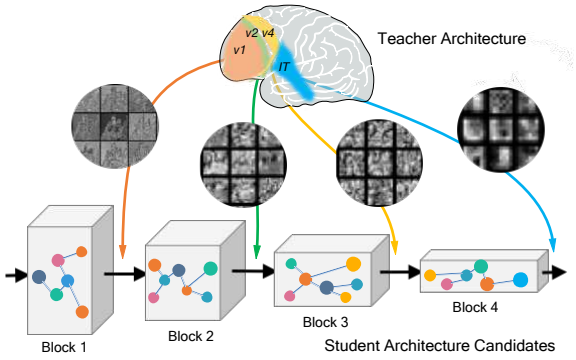

Figure 1. We consider a network architecture has several blocks, conceptualized as analogous to the ventral visual blocks $\mathcal{V}1$, $\mathcal{V}2$, $\mathcal{V}4$, and $\mathcal{IT}$ [27]. Then, we search for the candidate architectures (denoted by different nodes and paths) block-wisely guided by the architecture knowledge distilled from a teacher model.

1. Introduction

Due to the importance of automatically designing machine learning algorithms using machines, interest in the prospect of Automated Machine Learning (AutoML) has been a growing recently. Neural architecture search (NAS), as an essential task of AutoML, is expected to reduce the effort required to be expended by human experts in network architecture design. Research into NAS has been accelerated in the past two years by the industry, and a number of solutions have been proposed. However, the effectiveness and efficiency of existing NAS solutions are unclear. Typically, [26] and [35] even suggest that many existing solutions to NAS are no better than or struggle to outperform random architecture selection. Hence, the question of how to efficiently solve a NAS problem remains an active and unsolved research topic.

The most mathematically accurate solution to NAS is to train each of the candidate architectures within the search space from scratch to convergence and compare their performance; however, this is impractical due to the astonishingly high cost. A suboptimal solution is to train only the architectures in a search *sub-space* using advanced search strategies like Reinforcement Learning (RL) or Evolutionary Algorithms (EA); although this is still time-consuming,

as training even *one* architecture costs a long time (e.g., more than 10 GPU days for a ResNet on ImageNet). To speed up NAS, recent works have proposed that rather than training each of the candidates fully from scratch to convergence, different candidates should be trained concurrently by using shared network parameters. Subsequently, the ratings of different candidate architectures can be determined by evaluating their performance based on these undertrained shared network parameters. However, some latest works (e.g., [10] and [19]) have suggested that the evaluation based on the undertrained network parameters cannot correctly rank the candidate models, i.e., the architecture that achieves the highest accuracy cannot defend its top ranking when trained from scratch to convergence.

To address the above-mentioned issues, we propose a new solution to NAS where the search space is large, while the potential candidate architectures can be fully and fairly trained. We consider a network architecture that has several blocks, conceptualized as analogous to the ventral visual blocks $\mathcal{V}1$, $\mathcal{V}2$, $\mathcal{V}4$, and $\mathcal{IT}$ [27] (see Fig. 1). We then train each block of the candidate architectures separately. As guaranteed by the mathematical principle, the number of candidate architectures in a block reduces exponentially compared to the the number of candidates in the whole search space. Hence, the architecture candidates can be fully and fairly trained, while the representation shift caused by the shared parameters is reduced, leading to the correct candidate ratings. The correct and visiting-all evaluation improves the effectiveness of NAS. Moreover, thanks to the modest amount of the candidates in a block, we can even search for the depth of a block, which further improves the performance of NAS.

Moreover, lack of supervision for the hidden block creates a technical barrier in our greedy block-wise search of network architecture. To deal with this problem, we propose a novel knowledge distillation method, called **DNA**, that distills the neural architecture from an existing architecture. As Fig. 1 shows, we find that different blocks of an existing architecture have different knowledge in extracting different patterns of an image. For example, the lowest block acts like the $\mathcal{V}1$ of the ventral visual area, which extracts low-level features of an image, while the upper block acts like the $\mathcal{IT}$ area, which extracts high-level features. We also find that the knowledge not only lies, as the literature suggests, in the network parameters, but also in the network architecture. Hence, we use the block-wise representation of existing models to supervise our architecture search. Note that the performance of our searched architectures is not bounded by the performance of the supervising model. We have searched a number of architectures that have fewer parameters but significantly outperforms the supervising model, demonstrating the practicability of our DNA method.

Furthermore, inspired by the remarkable success of the transformers (e.g., BERT [12] and [31]) in natural language domain that discard the inefficient sequential training of RNN, we propose to parallelize the block-wise search in an analogous way. Specifically, for each block, we use the output of the previous block of the supervising model as the input for each of our blocks. Thus, the search can be sped up in a parallel way.

Overall, our contributions are three-fold:

- We propose to modularize the large search space of NAS into blocks, ensuring that the potential candidate architectures are fairly trained, and the representation shift caused by the shared parameters is reduced, which leads to correct ratings of the candidates. The correct and visiting-all ratings improve the effectiveness of NAS. Novelly, we also search for the depth of the architecture with the help of our block-wise search.
- We find that the knowledge of a network model lies not only, as the literature suggests, in the network parameters, but also in the network architecture. Therefore, we use the architecture knowledge from a teacher model to supervise our block-wise architecture search. Remarkably, the performance of our searched architecture has exceeded the teacher model, proving the practicability of our proposed DNA.
- Strong empirical results are obtained on ImageNet and CIFAR10. Typically, on ImageNet, our searched DNA-c achieves 1.5% higher top-1 accuracy over Efficient-B0 with the same model size; DNA-d achieves 78.4% top-1 accuracy, better than SCARLET-A (+1.5%) and ProxylessNAS (+3.3%) with lower parameter numbers. To the best of our knowledge, this is the state-of-the-art model in a mobile setting.

2. Related Work

Neural Architecture Search (NAS). NAS is hoped to replace the effort of human experts in network architecture design by machines. Early works [39, 3, 38, 7, 22] adopt an agent (e.g., an RNN or an EA method) to sample an architecture and get its performance through a complete training procedure. This type of NAS is computationally expensive and difficult to deploy on large-datasets. More recent studies [6, 21, 13, 1, 5] encode the entire search space as a weight sharing supernet. Gradient-based approches[21, 6, 33] jointly optimize the weight of the supernet and the architecture choosing factors by gradient descent. However, optimizing these choosing factors brings inevitable bias between sub-models. Since the sub-model performing poor in the beginning will get trained less and easily stay behind others, these methods depend heavily on the initial states, making it difficult to reach the best architecture. One-shot approaches [14, 10, 5, 4] ensure fairness among all sub-models. After training the supernet via path

dropout or path sampling, sub-models are sampled and evaluated with the weights inherited from the supernet. However, as identified in [4, 10, 19], there is a gap between the accuracy of the proxy sub-model with shared weights and the retrained stand-alone one. This gap narrows as the amount of weight sharing sub-models decrease [10, 19].

Knowledge Distillation. Knowledge distillation is a classical method of model compression, which aims at transferring knowledge from a trained *teacher* network to a smaller and faster *student* model. Existing works on knowledge distillation can be roughly classified into two categories. The first category is to use soft-labels generated by the teacher to teach the student, which is first proposed by [2]. Later, Hinton *et al*. [15] redefined knowledge distillation as training a shallower network to approach the teacher's output after the softmax layer. However, when the teacher model gets deeper, learning the soft-labels alone is insufficient. To address this problem, the second category of knowledge distillation proposes to employ the internal representation of the teacher to guide the training of the student [24, 37, 36, 32, 23]. [36] proposed a distillation method to train a student network to mimic the teacher's behavior in multiple hidden layers jointly. [32] proposed a progressive block-wise distillation to learn from teacher's intermediate feature maps, which eases the difficulty of joint optimization but increases the gap between the student and the teacher model. All existing works assume that the knowledge of a network model lies in the network parameter, while we find that the knowledge also lies in the network architecture. Moreover, in contrast to [32] , we proposed a parallelized distillation procedure to reduce both the gap and the time consumption.

3. Methodology

We begin with the inaccurate evaluation problem of existing NAS, based on which we define our block-wise search.

3.1. Challenge of NAS and our Block-wise Search

Let $\alpha \in \mathcal{A}$ and ω_α denote the network architecture and the network parameters, respectively, where $\mathcal{A}$ is the architecture search space. A NAS problem is to find the optimal pair $(\alpha^*, \omega_\alpha^*)$ such that the model performance is maximized. Solving a NAS problem often consists of two iterative steps, i.e., search and evaluation. A search step is to select an appropriate architecture for evaluation, while an evaluation step is to rate the architecture selected by the search step. The evaluation step is of most importance in the solution to NAS because an inaccurate evaluation leads to the ineffectiveness of NAS, and a slow evaluation results in the inefficiency of NAS.

Inaccurate Evaluation in NAS. The most mathematically accurate evaluation for a candidate architecture is to train it from scratch to convergence and test its performance, which, however, is impractical due to the awesome cost. For example, it may cost more than 10 GPU days to train a ResNet on ImageNet. To speed up the evaluation, recent works [4, 21, 6, 14, 33, 19] propose not to train each of the candidates fully from scratch to convergence, but to train different candidates concurrently by using shared network parameters. Specifically, they formulate the search space $\mathcal{A}$ into an over-parameterized supernet such that each of the candidate architecture α is a sub-net of the supernet. Let $\mathcal{W}$ denote the network parameters of the supernet. The learning of the supernet is as follows:

$$\mathcal{W}^* = \min_{\mathcal{W}} \mathcal{L}_{\text{train}}(\mathcal{W}, \mathcal{A}; \mathbf{X}, \mathbf{Y}), \tag{1}$$

where $\mathbf{X}$ and $\mathbf{Y}$ denote the input data and the ground truth labels, respectively. Here, $\mathcal{L}_{\text{train}}$ denotes the training loss. Then, the ratings of different candidate architectures are determined by evaluating their performance based on these shared network parameters, $\mathcal{W}^*$. However, as analyzed in Section 1, the optimal network parameter $\mathcal{W}^*$ does not necessarily indicate the optimal network parameters ω^* for the sub-nets (i.e., the candidate architectures) because the sub-nets are not fairly and fully trained. The evaluation based on $\mathcal{W}^*$ does not correctly rank the candidate models because the search space is usually large (e.g., $> 1e^{15}$). The inaccurate evaluation has led to the ineffectiveness of the existing NAS.

Block-wise NAS. [10] and [19] have suggested that when the search space is small, and all the candidates are fully and fairly trained, the evaluation could be accurate. To improve the accuracy of the evaluation, we divide the supernet into blocks of smaller sub-space. Specifically, Let $\mathcal{N}$ denote the supernet. We divide $\mathcal{N}$ into N blocks by the depth of the supernet and have:

$$\mathcal{N} = \mathcal{N}_N \dots \mathcal{N}_{i+1} \circ \mathcal{N}_i \cdots \circ \mathcal{N}_1, \tag{2}$$

where $\mathcal{N}_{i+1} \circ \mathcal{N}_i$ denotes that the $(i+1)$-th block is originally connected to the i-th block in the supernet. Then we learn each block of the supernet separately using:

$$\mathcal{W}_i^* = \min_{\mathcal{W}_i} \mathcal{L}_{\text{train}}(\mathcal{W}_i, \mathcal{A}_i; \mathbf{X}, \mathbf{Y}), \quad i = 1, 2 \cdots, N, \tag{3}$$

where $\mathcal{A}_i$ denote the search space in the i-th block. To make sure the weight sharing search space in the block-wise NAS is effectively reduced, the reduction rate is analysed as follows. Let d denote the depth of the i-th block and C denote the number of the candidate operations in each layer. Then the size of the search space of the i-th block is $C^{d_i}, \forall i \in [1, N]$; the size of the search space $\mathcal{A}$ is $\prod_{i=0}^{N} C^{d_i}$. This indicates a exponential reduction in the size of the weight-sharing search space:

$$\text{Reduction rate} = C^{d_i} / (\prod_{i=0}^{N} C^{d_i}). \tag{4}$$

In our experiment, the single weight-sharing search space in a block reduces significantly (e.g., Drop rate $\approx 1/(1e^{\frac{15}{N}})$), ensuring each candidate architecture $\alpha_i \in \mathcal{A}_i$ to be optimized sufficiently. Finally, the architecture is searched across the different blocks in the whole search space $\mathcal{A}$:

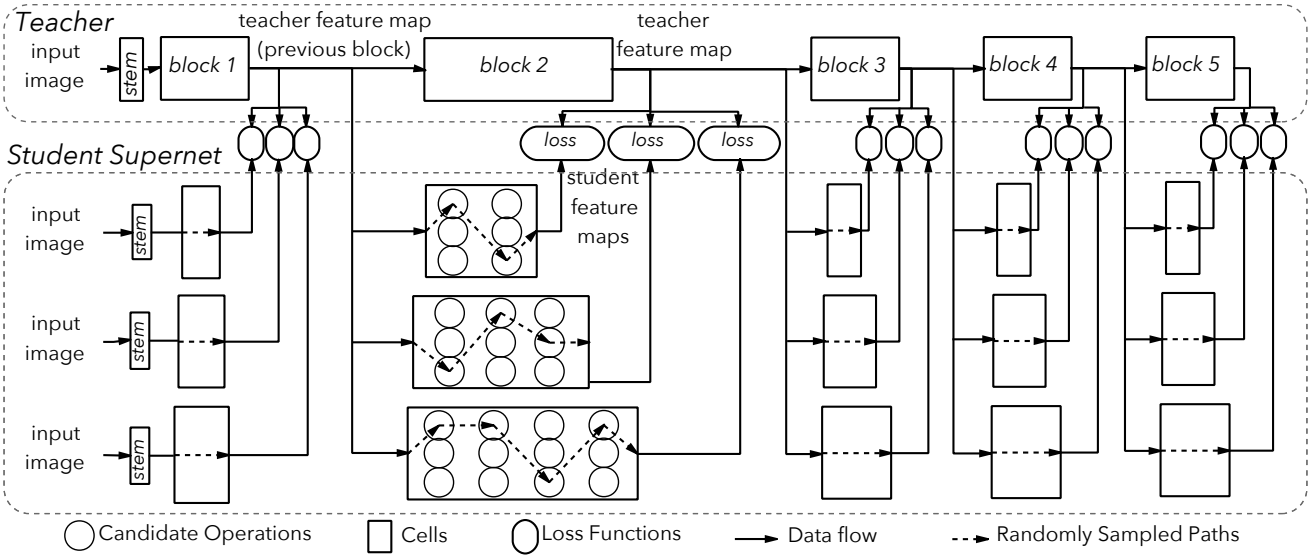

Figure 2. Illustration of our DNA. The teacher's previous feature map is used as input for both teacher and student block. Each cell of the supernet is trained independently to mimic the behavior of the corresponding teacher block by minimizing the l2-distance between their output feature maps. The dotted lines indicate randomly sampled paths in a cell.

$$\alpha^* = \arg\min_{\alpha \in \mathcal{A}} \sum_{i=1}^{N} \lambda_i \mathcal{L}_{\text{val}}(\mathcal{W}_i^*(\alpha_i), \alpha_i; \mathbf{X}, \mathbf{Y}), \tag{5}$$

where λ_i represents the loss weights. Here, $\mathcal{W}_i^*(\alpha_i)$ denotes the learned shared network parameters of the sub-net α_i and the supernet. Note that different from the learning of the supernet, we use the validation set to evaluate the performance of the candidate architectures.

3.2. Block-wise Supervision with Distilled Architecture Knowledge

Although we motivate well in Section 3.1, a technical barrier in our block-wise NAS is that we lack of internal ground truth in Eqn. (3). Fortunately, we find that different blocks of an existing architecture have different knowledge[1] in extracting different patterns of an image. We also find that the knowledge not only lies, as the literature suggests, in the network parameters, but also in the network architecture. Hence, we use the block-wise representation of existing models to supervise our architecture search. Let $\mathcal{Y}_i$ be the output feature maps of the i-th block of the supervising model (i.e., teacher model) and $\hat{\mathcal{Y}}_i(\mathcal{X})$ be the output feature maps of the i-th block of the supernet. We take L2 norm as the cost function. The loss function in Eqn. (3) can be written as:

$$\mathcal{L}_{\text{train}}(\mathcal{W}_i, \mathcal{A}_i; \mathbf{X}, \mathcal{Y}_i) = \frac{1}{K} \left\| \mathcal{Y}_i - \hat{\mathcal{Y}}_i(\mathcal{X}) \right\|_2^2, \tag{6}$$

where K denotes the numbers of the neurons in $\mathcal{Y}$. Moreover, inspired by the remarkable success of the transformers (e.g., BERT [12] and [31]) in natural language domain that discards the inefficient sequential training of RNN, we propose to parallelize the block-wise search in an analogous way. Specifically, for each block, we use the output $\mathcal{Y}_{i-1}$ of the $(i-1)$-th block of the teacher model as the input of the i-th block of the supernet. Thus, the search can be sped up in a parallel way. Eqn. (6) can be written as:

$$\mathcal{L}_{\text{train}}(\mathcal{W}_i, \mathcal{A}_i; \mathcal{Y}_{i-1}, \mathcal{Y}_i) = \frac{1}{K} \left\| \mathcal{Y}_i - \hat{\mathcal{Y}}_i(\mathcal{X}) \right\|_2^2, \tag{7}$$

Note that the performance of our searched architectures is not bounded by the performance of the supervising model, e.g., we have searched a number of architectures that have fewer parameters but significantly beats the supervising model. By scaling our architecture to the same model size as the supervising architecture, a more remarkable gain is further obtained, demonstrating the practicability of our DNA. Fig.2 shows a pipeline of our block-wise supervision with knowledge distillation.

3.3. Automatic Computation Allocation with Channel and Layer Variability

Automatically allocating model complexity of each block is especially vital when performing block-wise NAS under a certain constraint. To better imitate the teacher, the model complexity of each block may need to be allocated adaptively according to the learning difficulty of the corresponding teacher block. With the input image size and the stride of each block fixed, generally, the computation allocation is only related to the width and depth of each block, which are burdensome to search in a weight sharing supernet. Both the width and depth are usually pre-defined when designing the supernet for one-shot NAS methods. Most previous works include *identity* as a candidate operation to increase supernet scalability [4, 21, 6, 33, 19]. However, as pointed out in [8], adding identity as a candidate operation can bring difficulties for supernet convergence. In addition,

[1]The definition of knowledge is a matter of ongoing debate among philosophers. In this work, we specially define KNOWLEDGE as follows. **Knowledge** is the skill to recognize some patterns; **Parameter Knowledge** is the skill of using appropriate network parameter to recognize some patterns. **Architecture Knowledge** is the skill of using appropriate network structrue to recognize some patterns.

adding identity as a candidate operation may lead to a detrimental and unnecessary increase in the possible sequence of operations (e.g. sequence {conv, identity, conv} is equivalent to {conv, conv, identity}). This unnecessary increase of search space results in a drop of the supernet stability and fairness. Instead, [20] searches for layer numbers with fixed operations first, and subsequently searches for operations with a fixed layer number. However, the choice of operations is not independent from the layer number. To search for more candidate operations in this greedy way could lead to a bigger gap from the real target.

Thanks to our block-wise search, we can train several *cells* with different channel numbers or layer numbers independently in each stage to ensure channel and layer variability without the interference of identity operation, As shown in Figure 2, in each training step, the teacher's previous feature map is first fed to several cells (as suggested by the solid line), and one of the candidate operations of each layer in the cell is randomly chosen to form a path (as suggested by the dotted line). The weight of the supernet is optimized by minimizing the MSE loss with the teacher's feature map.

3.4. Searching for Best Student Under Constraint

Our typical supernet contains about 10^{17} sub-models, which stops us from evaluating all of them. In previous one-shot NAS methods, random sampling, evolutionary algorithms and reinforcement learning have been used to sample sub-models from the trained supernet for further evaluation. In most recent work [20, 19], a greedy search algorithm is used to progressively shrink the search space by selecting the top-performing partial models layer by layer. Considering our block-wise distillation, we propose a novel method to estimate the performance of all sub-models according to their block-wise performance and subtly traverse all the sub-models to select the top-performing ones under certain constraints.

Evaluation. In our method, we aim to imitate the behavior of the teacher in every block. Thus, we estimate the learning ability of a student sub-model by its evaluation loss in each block. Our block-wise search make it possible to evaluate all the partial models (about 10^4 in each cell). To accelerate this process, we forward-propagate a batch of input node by node in a manner similar to deep first search, with intermediate output of each node saved and reused by subsequent nodes to avoid recalculating it from the beginning. The feature sharing evaluation algorithm is outlined in Algorithm 1. By evaluating all cells in a block of the supernet, we can get the evaluation loss of all possible paths in one block. We can easily sort this list with about 10^4 elements in a few seconds with a single CPU. After this, we can select the top-1 partial model from every block to assemble a best student. However, we still need to find efficient models under different constraints to meet the needs of real-life applications.

Searching. After performing evaluation and sorting, the partial model rankings of each stage are used to find the best model under a certain constraint. To automatically allocate computational costs to each block, we need to make sure that the evaluation criteria are fair for each block. We notice that MSE loss is related to the size of the feature map and the variance of the teacher's feature map. To avoid any possible impact of this, a fair evaluation criterion, called relative $l1$ loss, is defined as:

$$\mathcal{L}_{\text{val}}(\mathcal{W}_i, \mathcal{A}_i; \mathcal{Y}_{i-1}, \mathcal{Y}_i) = \frac{\left\| \mathcal{Y}_i - \hat{\mathcal{Y}}_i(\mathcal{X}) \right\|_1}{K \cdot \sigma(\mathcal{Y}_i)}, \tag{8}$$

where $\sigma(\cdot)$ means standard deviation among all elements. All the $\mathcal{L}_{\text{val}}$ in each block of a sub-model is added up to estimate the ability to learn from the teacher. However, it is unnecessarily time-consuming to calculate the complexity and add up the loss for all 10^{17} candidate models. With ranked partial models in each block, a time-saving search algorithm (Alg. 2) is proposed to visit all possible models subtly. Note that we get the complexity of each candidate operation by a precalculated lookup table to save the time. The testing of next block is skipped if current partial model

Algorithm 1: Feature sharing evaluation

Input: Teacher's previous feature map G_{prev}, Teacher's current feature map G_{curr}, Root of the cell $Cell$, loss function $loss$
Output: List of evaluation loss L

```
define DFS-Forward(N, X):
    Y = N(X);
    if N has no child then
        append(L, loss(Y, G_curr));
    else
        for C in N.child do
            DFS-Forward(C, Y);
        end
    end
DFS-Forward(Cell, G_prev);
output L;
```

Algorithm 2: Traversal search

Input: Block index B, the teacher's current feature map G, constrain C, model pool list $Pool$
Output: best model M

```
define SearchBlock(B, size_prev, loss_prev):
    for i < length(Pool[B]) do
        size ← size_prev + size[i];
        if size > C then
            continue;
        end
        loss ← loss_prev + loss[i];
        if B is last block then
            if loss ≤ loss_best then
                loss_best ← loss;
                M ← index of each block
            end
            break;
        else
            SearchBlock(B + 1, size, loss);
        end
    end
SearchBlock(0);
output M;
```

Table 1. Our supernet design. "$l\#$" and "ch#" represents the layer and channel number of each cell.

model	teacher		student supernet					
block	$l\#$	ch#	$l\#$	ch#	$l\#$	ch#	$l\#$	ch#
1	7	48	2	24	3	24	2	32
2	7	80	2	40	3	40	4	40
3	10	160	2	80	3	80	4	80
4	10	224	3	112	4	112	4	96
5	13	384	4	192	5	192	5	160
6	4	640	1	320	-	-	-	-

combining with the smallest partial model in the following blocks already exceed the constraint. Moreover, it returns to the previous block after finding a model satisfying the constraint, to prevent testing of subsequent models with lower rank in current block.

4. Experiments

4.1. Setups

Choice of dataset and teacher model. We evaluated our method on ImageNet [11], a large-scale classification dataset that has been used to evaluate various NAS methods. During the architecture search, we randomly select 50 images from each class of the original training set to form a 50k-image validation set for the rating step of the NAS and use the remainder as the supernet training set. After that, all of our searched architectures are **retrained from scratch** on the original training set **without** supervision from the teacher network and tested on the original validation set. We further choose two widely used datasets, CIFAR-10 and CIFAR-100 [18], to test the transferability of our models.

We select EfficientNet-B7 [29] as our teacher model to guide our supernet training due to its state-of-the-art performance and relatively low computational cost comparing to ResNeXt-101 [34] and other manually designed models. We part the teacher model into 6 blocks by number of filters. The details of these blocks are presented in Table 1.

Search space and supernet design. We perform our search in two operation search spaces, both of which consist of variants of MobileNet V2's [25] Inverted Residual Block with Squeeze and Excitation [17]. We keep our first search space similar with most of the recent works [28, 29, 8, 9] to facilitate fair comparison in Section 4.2. We search among convolution kernel sizes of $\{3, 5, 7\}$ and expansion rates $\{3, 6\}$, six operations in total. For fast evaluation in Section 4.3 and 4.4, a smaller search space with four operations (kernel sizes of $\{3, 5\}$ and expansion rates $\{3, 6\}$) is used.

Upon operation search space, we further build a higher level search space to search for channel and layer numbers, as introduced in Section 3.3. We search among three cells in each of the first 5 blocks and one in the last block. The layer and channel numbers of each *cell* is shown in Table 1. The whole search space contains 2×10^{17} models.

Training details We separately train each *cell* in the supernet for 20 epochs under the guidance of teacher's feature map in corresponding block. We use 0.002 as start learning rate for the first block and 0.005 for all the other blocks. We use Adam as our optimizer and reduce the learning rate by 0.9 every epoch.

It takes 1 day to train a simple supernet (6 cells) using 8 NVIDIA GTX 2080Ti GPUs and 3 days for our extended supernet (16 cells). With the help of Algorithm 1, our evaluation cost is about 0.6 GPU days. To search for the best model under certain constraint, we perform Algorithm 2 on CPUs and the cost is less than one hour.

As for ImageNet retraining of searched models, we used the similar setting with [29]: batchsize 4096, RMSprop optimizer with momentum 0.9 and initial learning rate of 0.256 which decays by 0.97 every 2.4 epochs.

Table 2. Comparison of state-of-the-art NAS models on ImageNet. The input size is 224×224.

model	Params	FLOPS	Acc@1	Acc@5
SPOS [14]	-	319M	74.3%	-
ProxylessNAS [6]	7.1M	465M	75.1%	92.5%
FBNet-C [33]	-	375M	74.9%	-
MobileNetV3 [16]	5.3M	219M	75.2%	-
MnasNet-A3 [28]	5.2M	403M	76.7%	93.3%
FairNAS-A [10]	4.6M	388M	75.3%	92.4%
MoGA-A [9]	5.1M	304M	75.9%	92.8%
SCARLET-A [8]	6.7M	365M	76.9%	93.4%
PC-NAS-S [19]	5.1M	-	76.8%	-
MixNet-M [30]	5.0M	360M	77.0%	93.3%
EfficientNet-B0 [29]	5.3M	399M	76.3%	93.2%
random	5.4M	399M	75.7%	93.1%
DNA-a (ours)	4.2M	348M	77.1%	93.3%
DNA-b (ours)	4.9M	406M	77.5%	93.3%
DNA-c (ours)	5.3M	466M	77.8%	93.7%
DNA-d (ours)	6.4M	611M	**78.4%**	**94.0%**

4.2. Performance of searched models

As shown in Table 2, our DNA models achieve the state-of-the-art results compared with the most recent NAS models. Searched under a FLOPS constraint of 350M, DNA-a surpasses SCARLET-A with 1.8M fewer parameters. For a fair comparison with EfficientNet-B0, DNA-b and DNA-c are obtained with target FLOPS of 399M and parameters of 5.3M respectively. Both of them outperform B0 by a large margin (1.2% and 1.5%). Searched without constraint, our DNA-d achieves 78.4% top-1 accuracy with 6.4M parameters. When tested with the same input size (240×240) as EfficientNet-B1, DNA-d achieves 78.8% top-1 accuracy, being evenly accurate but 1.4M smaller than B1. MixNet-M, who uses the more efficient MixConv operation that we don't use, is 0.5% inferior to our smaller DNA-b. (See Appendix for details of our searched architecture)

Figure 3 compares the curve of **Accuracy vs. Parameters** and **Accuracy vs. FLOPS** for most recent NAS models. Our DNA models can achieve better accuracy with smaller model size and lower computation complexity than other most recent NAS models.

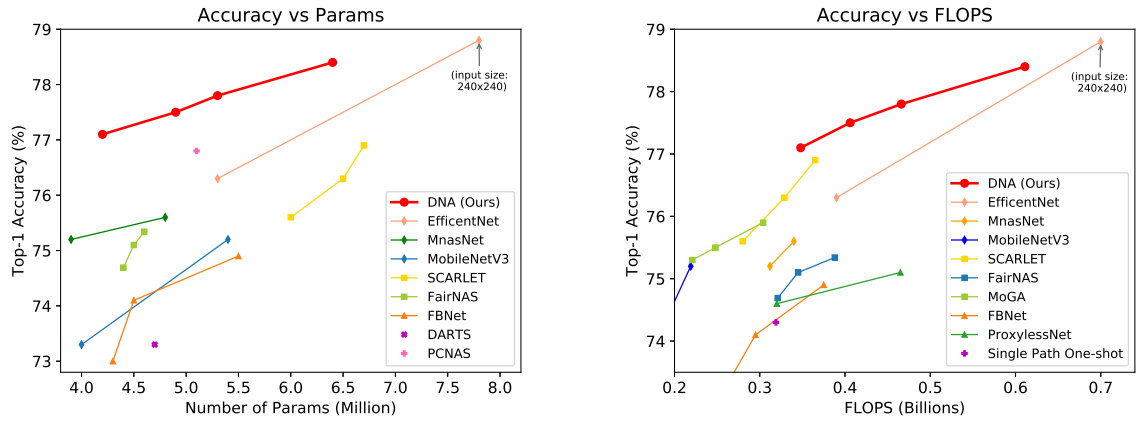

Figure 3. Trade-off of Accuracy-Parameters and Accuracy-FLOPS on ImageNet.

Table 3. Comparison of transfer learning performance of NAS models on CIFAR-10 and CIFAR-100. †: Re-implementation results with officially released models. Results within the parentheses are reported by the original paper.

Model	CIFAR-10 Acc	CIFAR-100 Acc
MixNet-M[30]	97.9%	87.4%
EfficientNet-B0	98.0% (98.1%)†	87.1% (88.1%)†
DNA-c (ours)	98.3%	88.3%

To test the transfer ability of our model, We evaluate our model on two widely used transfer learning datasets, CIFAR-10 and CIFAR-100. Our models maintain superiority after the transfer. The result is shown in Table 3.

4.3. Effectiveness

Model ranking. To evaluate the effectiveness of our NAS method, we compared the model ranking abilities between our method and SPOS (Single Path One-shot[14]) by visualizing the relationship between the evaluation metrics on proxy one-shot models and the actual accuracy of the stand-alone models. The two supernets are both 18 layers, with 4 candidate operations in each layer. The search space is described in Section 4.1. We trained our supernet with 20 epochs for each block, adding up to 120 epochs in total. The supernet of Single Path One-shot is also trained for 120 epochs as they proposed[14].

We sample 16 models from the search space and train them from scratch. As for model ranking test, we evaluate these sampled models in both supernets to get their predictive performance. The comparison of these two methods on model ranking is shown in Figure 4. Each of the sampled model has two corresponding points in the figure, representing the correlation between its predicted and true performance by two methods. Figure 4 indicates that SPOS barely rank the candidate models correctly because the subnets are not fairly and fully trained as analyzed in Section 3.1. While in our block-wise supernet, the predicted performance is highly correlated with the real accuracy of sampled models, which proves the effectiveness of our method.

Training progress. To analyse our supernet training process, we pick the intermediate models searched in every two training epochs (approximate to 5000 iterations) and retrain them to convergence. As shown in Figure 5, the accuracy of our searched models increase progressively as the training goes on until it converges between 16-th and 20-th epoch. It illustrates that the predictive metric of candidate models becomes more precise as the supernet converge. Note that the accuracy increase rapidly in the early stage with the same tendency of training loss decreasing, which evidences a correlation between accuracy of searched model and loss of supernet.

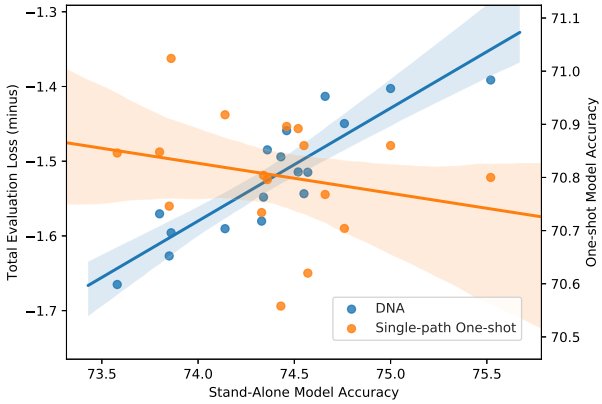

Figure 4. Comparison of ranking effectiveness for DNA and Single Path One-Shot[14]

Part of the teacher and student feature map of block 2 and 4 at epoch 16 is shown in Figure 6. As we can see, our student supernet can imitate the the teacher extraordinarily well. The textures are extremely close at every channel, even on highly abstracted 14×14 feature maps, proving the effectiveness of our distillation training procedure.

4.4. Ablation Study

Distillation strategy. We tested two progressive block-wise distillation strategy and compare their effectiveness with ours by experiments. All the three strategy is performed block by block by minimizing the MSE loss between feature maps of student supernet and the teacher. In strategy $S1$, the student is trained from scratch with all previous

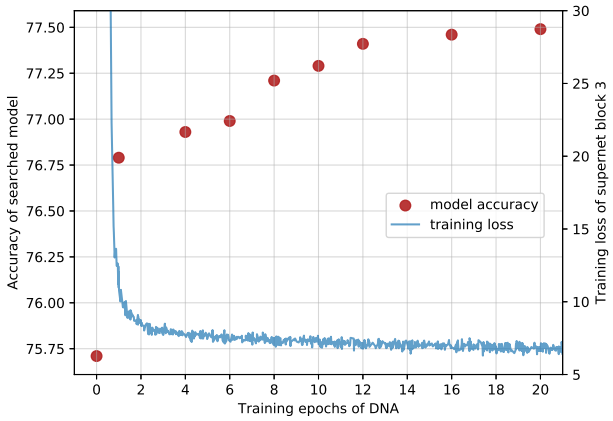

Figure 5. ImageNet accuracy of searched models and training loss of the supernet in training progress.

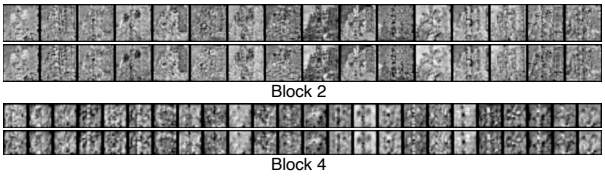

Figure 6. Feature map comparison between teacher (top) and student (bottom) of two blocks.

Table 4. Impact of each component of DNA. Our strategy is better than S1 and S2. Adding cells to increase channel and layer variability can boost performance of searched model both with and without constraint.

Strategy	Cell	Constrain	Params	Acc@1	Acc@5
S1			5.18M	77.0%	93.34%
S2			5.58M	77.15%	93.51%
Ours			5.69M	77.49%	93.68%
Ours	✓		6.26M	77.84%	93.74%
Ours		✓	5.09M	77.21%	93.50%
Ours	✓	✓	5.28M	77.38%	93.60%

Table 5. Comparison of DNA with different teacher. Note that all the searched models are retrained from scratch without any supervision of the teacher. †:EfficientNet-B7 is tested with 224×224 input size, to be consistent with distillation procedure.

Model	Params	Acc@1	Acc@5
EfficientNet-B0 (Teacher)	5.28M	76.3%	93.2%
DNA-B0	5.27M	77.8%	93.7%
EfficientNet-B7 (Teacher)	66M	77.8%†	93.8%†
DNA-B7	5.28M	77.8%	93.7%
DNA-B7-scale	64.9M	79.9%	94.9%

blocks in every stage. In strategy $S2$, the trained student parameters of the previous blocks is kept and freezed, thus those parameters are only used to generate the input feature map of current block. As discussed in Section 3.2, our strategy directly takes the teacher's previous feature map as input of the current block. The experimental results shown in Table 4 prove the superiority of our strategy.

Impact of multi-cell design. To test the impact of multi-cell search, we preform DNA with single cell in each block for comparison. As shown in Table 4, multi-cell search improves the top-1 accuracy of searched models by 0.2% under the same constraint (5.3M) and 0.3% for the best model in the search space without any constrain. Note that the single cell case of our method searched a model with lower parameters under the same constrain, this can be ascribed to the lower variability of channel and layer numbers.

Analysis of teacher-dependency. To test the dependency of DNA on the performance of teacher model, EfficientNet-B0 is used as the teacher model to search for a student in the similar size. The results is shown in Table 5. Surprisingly, performance of the model searched with EfficientNet-B0 is almost the same with the one searched with EfficientNet-B7, which means that the performance of our DNA method does not necessarily rely on high-performing teacher. Furthermore, the DNA-B0 outperforms its teacher by 1.5% with the same model size, which indicates that the performance of our architecture distillation may not be restricted by the performance of the teacher. Thus, we can improve the structure of any model by self-distillation architecture search. Thirdly, DNA-B7 achieves same top-1 accuracy with its $12.5\times$ heavier teacher; by scaling our DNA-B7 to the similar model size as the supervising architecture, a more remarkable gain is further obtained. The scaled student outperforms its heavy teacher by 2.1%, demonstrating the practicability of our DNA method.

5. Conclusion

In this paper, DNA, a novel architecture search method with block-wise supervision is proposed. We modularized the large search space into blocks to increase the effectiveness of one-shot NAS. We further designed a novel distillation approach to supervise the architecture search in a block-wise fashion. We then presented our multi-cell supernet design along with efficient evaluation and searching algorithms. We demonstrate that our searched architecture can surpass the teacher model and can achieve state-of-the-art accuracy on both ImageNet and two commonly used transfer learning datasets when trained from scratch without the helps of the teacher.

Acknowledgements

We thank DarkMatter AI Research team for providing computational resources. C. Li and X. Chang gratefully acknowledge the support of Australian Research Council (ARC) Discovery Early Career Researcher Award (DECRA) under grant no. DE190100626, Air Force Research Laboratory and DARPA under agreement number FA8750-19-2-0501. This work was also supported in part by the National Natural Science Foundation of China (NSFC) under Grant No.U19A2073, No.61976233, and in part by Natural Science Foundation of Guangdong Province under Grant No.2017A030312006.

References

[1] Youhei Akimoto, Shinichi Shirakawa, Nozomu Yoshinari, Kento Uchida, Shota Saito, and Kouhei Nishida. Adaptive stochastic natural gradient method for one-shot neural architecture search. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 171–180, 2019. 2

[2] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2654–2662, 2014. 3

[3] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2017. 2

[4] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 550–559, 2018. 2, 3, 4

[5] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Smash: one-shot model architecture search through hypernetworks. *International Conference on Learning Representations (ICLR)*, 2018. 2

[6] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *International Conference on Learning Representations (ICLR)*, 2019. 2, 3, 4, 6

[7] Yukang Chen, Gaofeng Meng, Qian Zhang, Shiming Xiang, Chang Huang, Lisen Mu, and Xinggang Wang. Renas: Reinforced evolutionary neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4787–4796, 2019. 2

[8] Xiangxiang Chu, Bo Zhang, Jixiang Li, Qingyuan Li, and Ruijun Xu. Scarletnas: Bridging the gap between scalability and fairness in neural architecture search. *arXiv preprint arXiv:1908.06022*, 2019. 4, 6

[9] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. Moga: Searching beyond mobilenetv3. *arXiv preprint arXiv:1908.01314*, 2019. 6

[10] Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. *arXiv preprint arXiv:1907.01845*, 2019. 2, 3, 6

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255, 2009. 6

[12] 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019. 2, 4

[13] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1761–1770, 2019. 2

[14] 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. *arXiv preprint arXiv:1904.00420*, 2019. 2, 3, 6, 7

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning Workshop*, 2014. 3

[16] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1314–1324, 2019. 6

[17] J Hu, L Shen, S Albanie, G Sun, and E Wu. Squeeze-and-excitation networks. *IEEE transactions on pattern analysis and machine intelligence*, 2019. 6

[18] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009. 6

[19] Xiang Li, Chen Lin, Chuming Li, Ming Sun, Wei Wu, Junjie Yan, and Wanli Ouyang. Improving one-shot nas by suppressing the posterior fading. *arXiv preprint arXiv:1910.02543*, 2019. 2, 3, 4, 5, 6

[20] Feng Liang, Chen Lin, Ronghao Guo, Ming Sun, Wei Wu, Junjie Yan, and Wanli Ouyang. Computation reallocation for object detection. *International Conference on Learning Representations (ICLR)*, 2020. 5

[21] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *International Conference on Learning Representations (ICLR)*, 2019. 2, 3, 4

[22] Renato Negrinho and Geoff Gordon. Deeparchitect: Automatically designing and training deep architectures. *International Conference on Learning Representations (ICLR)*, 2018. 2

[23] Nikolaos Passalis and Anastasios Tefas. Learning deep representations with probabilistic knowledge transfer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 268–284, 2018. 3

[24] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *International Conference on Learning Representations (ICLR)*, 2015. 3

[25] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018. 6

[26] Christian Sciuto, Kaicheng Yu, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. *International Conference on Learning Representations (ICLR)*, 2020. 1

[27] Stewart Shipp and Semir Zeki. Segregation of pathways leading from area v2 to areas v4 and v5 of macaque monkey visual cortex. *Nature*, 315(6017):322–324, 1985. 1, 2

[28] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2820–2828, 2019. 6

[29] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, pages 6105–6114, 2019. 6

[30] Mingxing Tan and Quoc V Le. Mixconv: Mixed depthwise convolutional kernels. In *Proceedings of the 30th British Machine Vision Conference (BMVC)*, 2019. 6, 7

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017. 2, 4

[32] Hui Wang, Hanbin Zhao, Xi Li, and Xu Tan. Progressive blockwise knowledge distillation for neural network acceleration. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2769–2775, 2018. 3

[33] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10734–10742, 2019. 2, 3, 4, 6

[34] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1492–1500, 2017. 6

[35] Antoine Yang, Pedro M Esperança, and Fabio M Carlucci. Nas evaluation is frustratingly hard. *International Conference on Learning Representations (ICLR)*, 2020. 1

[36] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4133–4141, 2017. 3

[37] Zhi Zhang, Guanghan Ning, and Zhihai He. Knowledge projection for deep neural networks. *arXiv preprint arXiv:1710.09505*, 2017. 3

[38] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2423–2432, 2018. 2

[39] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2017. 2