HourNAS: Extremely Fast Neural Architecture Search Through an Hourglass Lens

Zhaohui Yang[1,2], Yunhe Wang[2]*, Xinghao Chen[2], Jianyuan Guo[2],
Wei Zhang[2], Chao Xu[1], Chunjing Xu[2], Dacheng Tao[3], Chang Xu[3]
[1] Key Lab of Machine Perception (MOE), Dept. of Machine Intelligence, Peking University.
[2] Noah's Ark Lab, Huawei Technologies. [3] School of Computer Science, Faculty of Engineering, University of Sydney.
zhaohuiyang@pku.edu.cn; yunhe.wang@huawei.com; c.xu@sydney.edu.au

Abstract

Neural Architecture Search (NAS) aims to automatically discover optimal architectures. In this paper, we propose an hourglass-inspired approach (HourNAS) for extremely fast NAS. It is motivated by the fact that the effects of the architecture often proceed from the vital few blocks. Acting like the narrow neck of an hourglass, vital blocks in the guaranteed path from the input to the output of a deep neural network restrict the information flow and influence the network accuracy. The other blocks occupy the major volume of the network and determine the overall network complexity, corresponding to the bulbs of an hourglass. To achieve an extremely fast NAS while preserving the high accuracy, we propose to identify the vital blocks and make them the priority in the architecture search. The search space of those non-vital blocks is further shrunk to only cover the candidates that are affordable under the computational resource constraints. Experimental results on ImageNet show that only using 3 hours (0.1 days) with one GPU, our HourNAS can search an architecture that achieves a 77.0% Top-1 accuracy, which outperforms the state-of-the-art methods.

1. Introduction

In the past decade, progress in deep neural networks has resulted in the advancements in various computer vision tasks, such as image classification [6, 67, 7, 52], object detection [20, 41], and segmentation [25]. The big success of deep neural networks is mainly contributed to the well-designed cells and sophisticated architectures. For example, VGGNet [51] suggested the use of smaller convolutional filters and stacked a series of convolution layers for achieving higher performance, ResNet [26] introduced the residual blocks to benefit the training of deeper neural networks, and DenseNet [29] designed the densely connected blocks to stack features from different depths. Besides the efforts on the initial architecture design, extensive experiments [59, 24, 10] are often required to determine the weights and hyperparameters of the lightweight deep neural network [2, 11, 69, 35, 36, 37].

To automatically and efficiently search for neural networks of desireable properties (*e.g.*, model size and FLOPs) from a predefined search space, a number of Neural Architecture Search (NAS) algorithms [40, 65, 33, 70, 68, 18, 58, 61] have been recently proposed. Wherein, Evolutionary Algorithm (EA) based methods [48] maintain a set of architectures and generate new architectures using genetic operations like mutation and crossover. Reinforcement Learning (RL) based methods [75, 76] sample architectures from the search space and train the controllers accordingly. The differentiable based methods [40, 62, 63, 50] optimize the shared weights and architecture parameters, which significantly reduces the demand for computation resources and makes the search process efficient.

These methods have made tremendous efforts to greatly accelerate the search process of NAS. Nevertheless, given the huge computation cost on the large-scale dataset [62, 23, 64, 27, 3], *e.g.*, 9 GPU days for NAS on the ImageNet benchmark, most methods execute NAS in a compromised way. The architecture is first searched on a smaller dataset (*e.g.*, CIFAR-10 [32]), and then the network weight of the derived architecture is trained on the large dataset. An obvious disadvantage of this concession is that the performance of the selected architecture on the CIFAR-10 may not be well extended to the ImageNet benchmark [71]. We tend to use the minimal number of parameters without discrimination to construct an architecture that would achieve the maximal accuracy. But just as the popular 80-20 rule[1] goes, only a few parts could be critical to the architecture's success, and

*Corresponding author.

[1]The 80/20 rule (a.k.a Pareto principle, the law of vital few) is an aphorism that states, for many events, roughly 80% of the effects come from 20% of the causes.

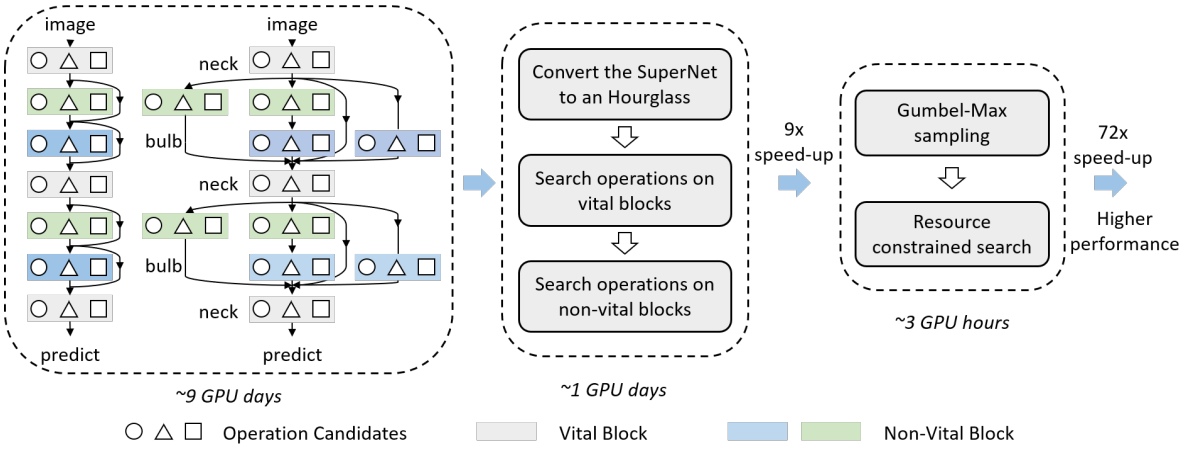


Figure 1. Blocks in the residual network are either "vital" or "non-vital", and they form the neck or bulb parts in the hourglass network. Two-stage search scheme speed up architecture search by 9× and the resource constrained search further accelerates architecture search by 72×.

we need to give them the most focus while balancing the parameter volume for other parts.

In this paper, we propose HourNAS for an accurate and efficient architecture search on the large-scale ImageNet dataset. Blocks in an architecture are not created equally. Given all the possible paths for the information flow from the input to the output of the network, blocks shared by these paths are *vital*, just as the neck of an hourglass to regulate the flow. We identify these vital blocks and make them the priority in the architecture search. The other non-vital blocks may not be critical to the accuracy of the architecture, but they often occupy the major volume of the architecture (like the bulb of an hourglass) and will carve up the resource budget left by the vital blocks. Instead of directly working in a large search space flooding with architectures that obviously do not comply with the constraints during deployment, we develop a space proposal method to screen out and optimize the most promising architecture candidates under the constraints. By treating the architecture search through an hourglass lens, the limited computation resource can be well allocated across vital and non-vital blocks. We design toy experiments on residual networks to illustrate the varied influence of vital and non-vital blocks on the network accuracy. The resulting HourNAS costs only about 3 hours (0.1 GPU days) on the entire ImageNet dataset to achieve a 77.0% Top-1 accuracy, which outperforms the state-of-the-art methods.

2. Related Works

This section reviews the methods for neural architecture search algorithms. Then, we discuss layer equality, *i.e.*, the importance of different blocks in deep neural networks.

Neural Architecture Search. To automate the design of neural models, neural architecture search (NAS) was introduced to discover efficient architectures with competitive performance. Reinforcement learning (RL) and evolution algorithm (EA) were widely adopted in NAS [76, 75, 42, 48, 55, 23, 21]. However, these methods were highly computationally demanding. Many works have been devoted to improving the efficiency of NAS from different perspectives, *e.g.*, by adopting the strategy of weight sharing [45] or progressive search [39]. Differentiable based NAS attracted great interest as it drastically reduces the searching cost to several days or even hours [62, 40, 16, 63, 46, 38, 22, 8, 34, 74]. For example, DARTS [40] adopted the continuous architecture representation to allow efficient architecture search via gradient descent. Meanwhile, as discusses in TuNAS [1], gradient-based NAS methods consistently performed better than random search, showing the power of searching for excellent architectures. However, the most efficient gradient-based NAS methods still took dozens of days for directly searching on target large-scale dataset (*e.g.*, ImageNet). Thus, an efficient method for directly searching deep neural architectures on large-scale datasets and search spaces is urgently required.

Layer Equality. Most of existing NAS methods treated all layers with the same importance during the search. However, convolution neural networks are always over-parameterized and the impact on the final accuracy of each layer is totally different. Zhang *et al.* [72] re-initialized and re-randomized the pre-trained networks, and found that some layers are robust to the disturbance. For some intermediate layers, the re-initialization and re-randomization steps did not have negative consequences on the accuracy. Veit *et al.* [60] decomposed residual networks and found that skipping some layers does not decrease the accuracy significantly. Ding *et al.* [15] pruned different layers separately and found some layers are more important to the final accuracy. It is obvious that the layers are not created equal, and some layers are more important than others.

In this paper, we analyze the causes of the inequality

phenomenon in the residual network and exploit this property in neural architecture search to improve its efficiency.

3. Hourglass Neural Architecture Search

In this section, we revisit the neural architecture search from an hourglass way. The vital few blocks should be searched with a higher priority to guarantee the potential accuracy of the architecture. The non-vital blocks that occupy the major volume are then searched in an extremely fast way by focusing on the discovered space proposals.

3.1. Vital Blocks: the Neck of Hourglass

In this paper, we focus on the serial-structure NAS SuperNet [55, 56, 62, 4, 54, 53], as it is hardware-friendly and capable of achieving superior performance. Before we illustrate vital blocks in a general NAS, we first take ResNet [26] as an example for the analysis. ResNet is one of the most popular manually designed architectures. It is established by stacking a series of residual blocks, and the residual block is defined as,

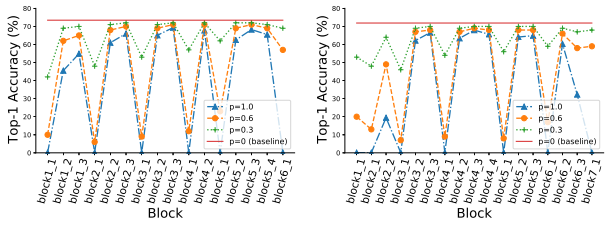
$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \mathbf{w}) + \mathbf{x}, \tag{1}$$

where $\mathbf{x}$ is the input feature map, $\mathcal{F}$ denotes the transformation (*e.g.*, convolution and batch normalization for vision tasks) and $\mathbf{w}$ stands for the trainable parameters.[2] From the information flow perspective, there are two paths to transmit the information from the node $\mathbf{x}$ to the node $\mathbf{y}$, *i.e.*, the shortcut connection and the transformation $\mathcal{F}$. If there are m residual blocks in a network, there will be 2^m different paths for the information propagation in total. A general neural network $\mathcal{N}$ based on the residual blocks [26, 55, 62] can therefore be approximated as the ensemble of a number of paths [60] $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$, *i.e.*, $\mathcal{N}(X) \approx \sum_{i=1}^{n} \mathcal{P}_i(X)$, where each path $\mathcal{P}_i$ is set up by a series of blocks, X is the input data, and n is the number of all the paths.

It is worth noticing that there are a few blocks existing in all the possible paths, *e.g.*, the gray blocks in Fig. 1. These self-contained blocks do not participate in forming any residual blocks, but they are *vital*, because of their appearance in every path from the input to the output of the network. On the other hand, the green and blue blocks in Fig. 1 are a part of the residual blocks $\mathbf{y} = \mathcal{F}(\mathbf{x}, \mathbf{w}) + \mathbf{x}$, where the information can be transmitted through the plain transformation $\mathcal{F}(\mathbf{x}, \mathbf{w})$ or the shortcut connection $\mathbf{x}$ to the next block, so they are not that vital.

Identify and Examine Vital Blocks. Given the paths $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ in a general residual network $\mathcal{N}$, the vital blocks shared by all the paths can be identified through $\hat{\mathcal{P}} = \mathcal{P}_1 \cap \cdots \cap \mathcal{P}_n$, where $\mathcal{P}_i \cap \mathcal{P}_j$ denotes the intersection set of those blocks in paths $\mathcal{P}_i$ and $\mathcal{P}_j$. In the popular residual networks, such as ResNet [26] and FBNet [62], the vital blocks are exactly the first convolution layer, the last fully connected layer, the downsampling blocks, and the channel expansion blocks. These vital blocks are critical to the accuracy of the whole architecture, as they exist in all paths and act as the neck of the hourglass to control the information flow. In contrast, the other blocks would always find substitutes for themselves to keep the information flow, and they thus play a secondary role in the whole architecture.

[2] As for the downsample blocks (reduce the feature map size) and the channel expansion blocks (increase the channel number), we follow [60] and use $\mathbf{y} = \mathcal{F}(\mathbf{x}, \mathbf{w})$ to express.



(a) MnasNet (b) MobileNetV2

Figure 2. The diagram of block importance by using the MnasNet and MobileNetV2 pretrained models.

We further take mobile architectures as an example to illustrate the different influence of vital and non-vital blocks on the network accuracy. A random mask function $\mathcal{M}(\mathbf{y}, p)$ is introduced to destroy the output of blocks in the pretrained MnasNet [55] and MobileNetV2 [49], where $\mathbf{y}$ is the output feature map, and $0 \leq p \leq 1$ is the probability. In particular, every channel is reset to 0 with a probability p. Fig. 2 shows the accuracy change of MnasNet [55] and MobileNetV2 [49] resulting from the feature distortion on different blocks. The $blocki\_j$ denotes the j-th block in the i-th stage. As discussed above, the first block in every stage is the vital block. We set $p = \{0.3, 0.6, 1.0\}$ to gradually increase the degree of feature distortion. Each time we only manipulate one block while keeping others unchanged in the network. For the non-vital blocks (*e.g.*, block3_2 and block3_3 in both MnasNet and MobileNetV2), even if all the channels are reset to zero (*i.e.*, p=1.0), the network does not undergo a significant accuracy degradation. However, a small portion (p=0.3) of channels that are masked out for those vital blocks (*e.g.*, block1_1 and block2_1) will lead to an obvious accuracy drop.

Revisit Neural Architecture Search. The goal of NAS is to search for the architecture of a higher accuracy under the constraints of available computational resources. In other words, NAS can be regarded as a problem of resource allocation. Vital blocks are potentially the most important and need to be put as the priority. As a result, more resources are better to be first allocated to the vital blocks, and the remaining resources are used for the non-vital blocks design. This therefore naturally motivates us to develop a two-stage search algorithm. During the first stage, we construct the minimal SuperNet $\mathcal{S}_{vital}$ by stacking all the vital layers and search the vital blocks. The weights and architecture param-

eters are optimized alternatively in a differentiable way [62]. In the second stage, we fix the derived architecture of those vital blocks, and allocate the computational resources to search for the non-vital blocks.

3.2. Non-Vital Blocks: the Bulb of Hourglass

Non-vital blocks are often composed of a large number of parameters. They look like the bulb of the hourglass to determine the whole volume size. If the computational resources are unlimited to deploy the neural network, then we can make the network wider and deeper for achieving a higher performance [26, 56]. However, the searched architectures are to be deployed on mobile devices which have demanding constraints of the resources, *e.g.*, model size, and FLOPs. Without investigating these constraints, it would be ineffective to directly sample the architecture from a large search space. For example, if the sampled architectures cannot fully use the available computation resource, the resulting models might perform poorer than expected, which has been analyzed by a number of multi-objective NAS works [42, 66, 19, 55] (the Pareto front). Otherwise, if the sampled architectures overwhelm the use of computation resources, they would be difficult to be deployed. To tackle the dilemma, we introduce an efficient sampling operation to avoid wasting too much time on search unimportant operations, and a space proposal strategy to put more attention on architectures that meet the target computational resources.

3.3. Space Proposal for Efficient Resource Constrained Search

A general differentiable neural architecture search algorithm can be formulated as a bilevel optimization problem [40]:

$$\begin{aligned} \theta^* &= \arg\min_{\theta} \mathcal{H}_{val}(w^*(\theta), \theta), \\ \text{s.t. } w^*(\theta) &= \arg\min_{w} \mathcal{H}_{train}(w, \theta), \end{aligned} \tag{2}$$

where $\mathcal{H}$ is the cross-entropy loss function, and θ denotes the architecture parameters. If the accuracy is the only objective to be considered for searching, a complex architecture would be preferred for achieving a highest accuracy (see Sec. 4.3). However, if the obtained architectures are to be deployed on mobile devices, we may always have the computational resource constraints from the environment . Thus, neural architecture search that considers the multiple objectives can be formulated as,

$$\begin{aligned} \theta^* &= \arg\min_{\theta} \mathcal{H}_{val}(w^*(\theta), \theta) + \alpha \times \mathcal{T}(\theta), \\ \text{s.t. } w^*(\theta) &= \arg\min_{w} \mathcal{H}_{train}(w, \theta), \end{aligned} \tag{3}$$

where $\mathcal{T}(\theta)$ is the regularization term that encourages the produced architecture to satisfy the target computational resource constraints. Assuming the constraints (targets) on computational resources (*e.g.*, model size, FLOPs) are $T_{i\in\{1,\dots,n\}}$, where n is the number of objectives, an efficient and controllable way is to initialize architectures that satisfy T. Thus, we introduce the concept of space proposal. The space proposal is a subspace of the large search space, and all the sampled architectures from the space proposal satisfy the target resources. As a result, the search phase would not waste resources on optimizing useless architectures. In addition, the space proposal ensures "what you set is what you get". Similar to gradient-based NAS, the space proposal is represented by the architecture parameters.

We take the FLOPs as an example to describe how to optimize a space proposal. Suppose θ represents the trainable architecture parameters of the NAS SuperNet and the size is $L \times O$, where L is the maximum depth and O is the number of candidate operations in each layer. A number of methods $\mathcal{G}$ are capable of sampling architectures A_θ from architecture parameters θ,

$$A_\theta = \mathcal{G}(\theta), \quad \sum_o A_{l,o} = 1, \tag{4}$$

where $\mathcal{G}$ is usually specified as softmax [40], Gumbel-softmax [62], Gumbel-Max [17, 5], *etc*. The $A_{l,o}$ is the o-th operation in the l-th layer.

The FLOPs table F of the SuperNet $\mathcal{S}$ is of size $L \times O$, where $F_{l,o}$ denotes the FLOPs of the o-th operation in the l-th SuperBlock. The FLOPs for sampled architecture A_θ is calculated as $\mathcal{R}_F(A_\theta) = \text{sum}(A_\theta \odot F)$, where $\odot$ is the element-wise product. Assuming the target FLOPs is T_F, the optimization is formulated as,

$$\theta^F = \arg\min_{\theta} |\mathcal{R}_F(\mathcal{G}(\theta)) - T_F| / M_F, \tag{5}$$

where M_F is a constant scalar denotes the maximum FLOPs of the sampled architectures, and this term is used for normalizing the objective to $[0, 1]$. We extend Eqn 5 to n different objectives. The targets for n objectives are $T_{i\in\{1,\dots,n\}}$, and the optimization is defined as,

$$\theta^T = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} |\mathcal{R}_i(\mathcal{G}(\theta)) - T_i| / M_i, \tag{6}$$

which $\mathcal{R}_i(\mathcal{G}(\theta))$ is the resource demand of the architecture sampled by $\mathcal{G}(\theta)$ on the i-th objective.

This optimization problem is easily to be solved in a few seconds. The solution θ^T can be regarded as a space proposal under the constraints T, and the structure A_{θ^T} sampled from θ^T by Eqn 4 would be more easily to satisfiy target resources T. Instead of relying on a single optimal solution θ^T, we turn to an ensemble way to start from different random initializations and derive a series of space proposals $\Theta^T = \{\theta_1^T, \cdots, \theta_m^T\}$, where m is the number of space

proposals. The orthogonal constraint is also introduced to further increase the diversity of different space proposals, which is formulated as,

$$\theta_1, \ldots, \theta_m = \arg\min_{\theta_1,\ldots,\theta_m}\Big(\frac{1}{nm}\sum_{j=1}^{m}\sum_{i=1}^{n}|\mathcal{R}_i(\mathcal{G}(\theta_j)) - T_i|/M_i + \beta \times \sum(|O - I|)\Big), \tag{7}$$

where $O = \Theta\Theta^T$ is an $m \times m$ matrix and element $O_{i,j}$ denotes the inner product of θ_i and θ_j. The term $\sum(|O - I|)$ regularizes m space proposals to be orthogonal, which indicates that the architectures sampled from different space proposals are different. A uniformly initialized auxiliary parameter Π of size m is then introduced to sample space proposals,

$$\pi = \mathcal{P}(\Pi), \quad \sum_i \pi_i = 1, \tag{8}$$

where $\mathcal{P}$ could be softmax, Gumbel-softmax or Gumbel-Max, π is the sampled vector from Π that used for combine the architectures sampled from m space proposals, and the ensembled architecture A_Θ that used for updating the SuperNet is defined as,

$$A_\Theta = \sum_i \pi_i \cdot A_{\theta_i} = \sum_i \pi_i \cdot \mathcal{G}(\theta_i), \tag{9}$$

where A_Θ is utilized for updating network parameter w on train set $\mathcal{D}_{train}$ and architecture parameter Π, Θ on validation set $\mathcal{D}_{val}$, respectively. Every space proposal θ_i optimizes towards the good architectures in the space proposal and Π optimizes towards better proposals. The NAS framework by using the space proposal strategy is summarized as,

$$\Pi^*, \Theta^* = \arg\min_{\Pi,\Theta} \mathcal{H}_{val}(w^*(\Pi,\Theta), \Pi, \Theta) + \alpha \times \mathcal{T}(\Pi, \Theta),$$
$$\text{s.t. } w^*(\Pi,\Theta) = \arg\min_{w} \mathcal{H}_{train}(w, \Pi, \Theta), \tag{10}$$

where $\mathcal{T}$ (Eqn 6) is the regularization on space proposal parameters (Eqn 7), and α is the slope of the multi-objective loss term.

3.4. Overall Search Algorithm

Based on the proposed method, we summarize the overall search algorithm in Alg. 1.

Algorithm 1 The searching algorithm of HourNAS.

Input: The NAS supernet $\mathcal{S}$, the computational targets $T_{i\in\{1,\ldots,n\}}$, the train set D_{train} and validation set D_{val}, the searching epochs for vital blocks E_{vital} and non-vital blocks $E_{n-vital}$, the number of space proposals m, iterations I_{sp} for training space proposals.

// Search Vital Blocks
Constructing the minimal SuperNet $\mathcal{S}_{vital}$ by stacking all the vital layers and the architecture parameter θ_{vital}.
for e = 1 to E_{vital} **do**
  for data and target pair (X_{tr}, Y_{tr}) in D_{train} **do**
    Sample network A from θ_{vital}, calculate loss and update network parameters.
  end for
  for data and target pair (X_{val}, Y_{val}) in D_{val} **do**
    Sample network A from θ_{vital}, calculate loss and update θ_{vital}.
  end for
end for
The operations with the highest importance are selected to form the vital layers.
// Optimize m space proposals
According to the computational targets T, HourNAS optimizes m proposals $\Theta^T = \{\theta_1^T, \cdots, \theta_m^T\}$ for I_{sp} iterations (Eqn 7), and construct the proposal sampler π (Eqn 8).
// Search Non-Vital Blocks
for e = 1 to $E_{n-vital}$ **do**
  for data and target pair (X_{tr}, Y_{tr}) in D_{train} **do**
    Sample network A from π and Θ, calculate loss and update network parameters.
  end for
  for data and target pair (X_{val}, Y_{val}) in D_{val} **do**
    Sample network A from π and Θ, calculate loss and update π and Θ.
  end for
end for
Fix operations by selecting the space proposal and operations with the highest probability.

Output: The architecture A which satisfies the computational targets $T_{i\in\{1,\ldots,n\}}$.

4. Experiments

In this section, we first describe the experimental settings and then extensively evaluate the proposed HourNAS on several popular NAS search spaces [62, 55, 56] on ImageNet. The models trained on CIFAR-10 dataset by using MindSpore toolkit are available at https://gitee.com/mindspore/mindspore/tree/master/model_zoo/research/cv/HourNAS

4.1. Experimental Settings

We use the HourNAS to search on the complete ImageNet train set. The subset $\mathcal{D}_{tr}$ takes 80% of the train set to train network parameters and the rest $\mathcal{D}_{val}$ is used to update architecture parameters. We search on three popular search spaces, *i.e.*, FBNet [62], MnasNet [55], and EfficientNet [56]. For any of our searched architecture, the training strategy is the same as the corresponding baseline. We use the NVIDIA V100 GPU to measure the search time

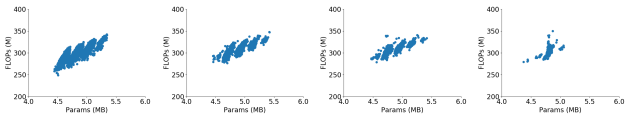
and compare with previous works fairly. The V100 GPU is also adopted by a number of literatures, for example, PDARTS [9], FBNet [62].

The HourNAS first searches the vital blocks for one epoch (about 1 hour). Then, HourNAS optimizes multiple space proposals according to the computational targets and searches the non-vital blocks for one epoch (about 2 hours). The whole searching process requires only one V100 GPU within 3 hours. Extending the search time will not further improve the accuracy, because the distribution of architecture parameters is stable. For competing methods like MnasNet [55], 3 GPU hours could only train one sampled architecture and the RL controller has no difference with random search. We utilize the Gumbel-Max [44, 5, 17] to sample operations according to the learned importance, which avoids wasting searching time on undesired operations. Gumbel-Max samples an operation according to the learned probability distribution (*i.e.*, importance). The sampling frequencies of those poor operations tend to be relatively low, so that we can effectively reduce the time spent on them. The Gumbel-Max sampling accelerates every iteration by around O times, where O is the number of candidate operations in every layer. During searching, we follow FBNet [62] and add the temperature τ (Eqn 4) for sharpening the distribution progressively. The temperature τ starts from 5.0 and multiply 0.9999 at every iteration. Slope parameter α, β are emperically set to 5.0 and 1e-2, respectively. Learning rates for optimizing weights and architecture parameters are 0.1 and 0.01, respectively. Adam [31] optimizer is used to update architecture parameters.

4.2. Comparison with State-of-the-arts

FBNet Search Space (FBNetSS). We first evaluate our HourNAS on the popular FBNet search space (FBNetSS), which contains $9^{22} \approx 1 \times 10^{21}$ architectures. HourNAS first searches the vital blocks and the results show that operations with the expansion ratio of six have significantly higher probabilities than other operations. We choose the operations with the highest probabilities to form the vital blocks. This result is in line with our intuition that the complex operations have the greatest feature extraction ability on the large dataset, *i.e.*, ImageNet.

After fixing the operations of the vital blocks, we are interested in finding the appropriate proposal number m. We set the computational resources the same as FBNet-B and search architectures using m different space proposals. We first visualize the distribution of sampled architectures from the space proposal by gradually decreasing the temperature τ. Each space proposal is optimized for 1000 iterations in total (Eqn 6). The computational targets are set to 4.8M parameters (x-axis) and 300M FLOPs (y-axis), which are consistent with FBNet-B [62]. As shown in Fig. 3, with the decrease of temperature of τ, the sampled architectures



(a) $\tau = 5.0$ (b) $\tau = 1.0$ (c) $\tau = 0.5$ (d) $\tau = 0.1$

Figure 3. The distribution of 10,000 architectures sampled from optimized space proposal under different temperatures τ.

satisfy the computational targets more precisely.

The optimization step for constructing several space proposals takes only a few seconds, which is an efficient solution for controllable multi-objective neural architecture search. We enumerate $m = \{1, 2, 4, 8, 16\}$ and the operations with the highest probability according to Π and Θ are selected after searching. The architectures are evaluated on the CIFAR-10 dataset to determine the appropriate space proposal number m for finding superior architectures. In retraining, we integrate CutOut [14] to make networks generalize better. The stochastic depth [30] is not used. As shown in Tab. 2, $m = 8$ could result in a well-performed architecture and we use $m = 8$ in the following experiments to achieve a better trade-off between searching costs and performance.

Table 2. Comparison of image classifiers on CIFAR-10 dataset.

Model	Test Error (%)	Params (M)	Search Cost (GPU days)
HourNAS (m=1)	3.86	2.8	0.1
HourNAS (m=2)	3.54	2.8	0.1
HourNAS (m=4)	3.41	2.7	0.1
HourNAS (m=8)	3.39	2.8	0.1
HourNAS (m=16)	3.37	2.8	0.1

We search for models on the ImageNet dataset, *i.e.*, HourNAS-FBNetSS-A and HourNAS-FBNetSS-B. To fairly compare with FBNet, HourNAS-FBNetSS-A has the same model size and FLOPs as FBNet-B, and HourNAS-FBNetSS-B has the same computational requirements as FBNet-C. We train the networks for 350 epochs in total with a batch size of 512. Learning rate starts from 0.25 and the weight decay is 1e-5. Label smoothing and learning rate warmup strategies are also used. The activation after convolution is the ReLU function. The training strategy is the same as FBNet [62] without using bells and whistles. As shown in Tab. 1, the HourNAS-FBNetSS-A and HourNAS-FBNetSS-B achieve competitive accuracies with FBNet-B and FBNet-C, and the search time is drastically reduced by two orders of magnitude. We search HourNAS-FBNetSS-A for three times using different random seeds, and the standard deviation of Top-1 accuracy is 0.1%.

Enlarged FBNet Search Space (EFBNetSS). MixConv [57] indicates that a larger kernel size leads to better performance. To understand the impact of search space and to further verify the effectiveness of HourNAS, we

Table 1. Overall comparison on the ILSVRC2012 dataset.

Model	Type	Search Dataset	Search Cost (GPU days)	Params (M)	FLOPS (M)	Top-1 (%)	Top-5 (%)
ResNet50 [26]	manual	-	-	25.6	4100	75.3	92.2
MobileNetV1 [28]	manual	-	-	4.2	575	70.6	89.5
MobileNetV2 [49]	manual	-	-	3.4	300	72.0	91.0
MobileNetV2 (1.4×)	manual	-	-	6.9	585	74.7	92.5
ShuffleNetV2 [43]	manual	-	-	-	299	72.6	-
ShuffleNetV2 (1.5×)	manual	-	-	3.5	299	72.6	-
FPNASNet [13]	auto	CIFAR-10	0.8	3.4	300	73.3	-
SNAS (mild) [63]	auto	CIFAR-10	1.5	4.3	522	72.7	90.8
AmoebaNet-A [48]	auto	CIFAR-10	3150	5.1	555	74.5	92.0
PDARTS [9]	auto	CIFAR-10	0.3	4.9	557	75.6	92.6
NASNet-A [76]	auto	CIFAR-10	1800	5.3	564	74.0	91.3
GDAS [17]	auto	CIFAR-10	0.2	5.3	581	74.0	91.5
PNAS [39]	auto	CIFAR-10	225	5.1	588	74.2	91.9
CARS-I [66]	auto	CIFAR-10	0.4	5.1	591	75.2	92.5
DARTS [40]	auto	CIFAR-10	4	4.9	595	73.1	91.0
MdeNAS [73]	auto	CIFAR-10	0.2	6.1	-	74.5	92.1
RCNet [64]	auto	ImageNet	8	3.4	294	72.2	91.0
SPOSNAS [23]	auto	ImageNet	13	5.3	465	74.8	-
ProxylessNAS [4]	auto	ImageNet	8.3	7.1	465	75.1	92.5
FBNet-B [62]	auto	ImageNet	9	4.8	295	74.1	-
FBNet-C [62]	auto	ImageNet	9	5.5	375	74.9	-
HourNAS-FBNetSS-A	auto	ImageNet	**0.1**	4.8	298	**74.1**	**91.8**
HourNAS-FBNetSS-B	auto	ImageNet	**0.1**	5.5	406	**75.0**	**92.2**
HourNAS-EFBNetSS-C	auto	ImageNet	**0.1**	4.8	296	**74.1**	**91.6**
HourNAS-EFBNetSS-D	auto	ImageNet	**0.1**	5.5	394	**75.3**	**92.3**
MnasNet-A1 [55]	auto	ImageNet	3800	3.9	312	75.2	92.5
HourNAS-MnasNetSS-E	auto	ImageNet	**0.1**	3.8	313	**75.7**	**92.8**
EfficientNet-B0 [56]	auto	ImageNet	-	5.3	390	76.8	-
HourNAS-EfficientNetSS-F	auto	ImageNet	**0.1**	5.3	383	**77.0**	**93.5**

slightly enlarge the search space of FBNet. We add the blocks with kernel size $k = 7$ and remove the blocks with group $g = 2$. This modification results in a search space containing 1×10^{22} architectures, which is 10 times larger than the original one. The multi-objectives are the same as HourNAS-FBNetSS-A and HourNAS-FBNetSS-B. We list the searched architectures in Tab. 1. The HourNAS-EFBNetSS-C achieves the same Top-1 accuracy with HourNAS-FBNetSS-A and HourNAS-EFBNetSS-D surpasses HourNAS-FBNetSS-B by 0.3% Top-1 accuracy. The larger kernel size $k = 7$ ensures that the architectures are capable of perceiving the characteristics of a larger area.

MnasNet Search Space (MnasNetSS). We further apply our proposed HourNAS to the search space of MnasNet [55]. The search space contains 2.5×10^{23} architectures in total and is larger than FBNet search space. We select MnasNet-A1 as the baseline and use its number of the parameters and FLOPs as two objectives to optimize 8 space proposals. The discovered HourNAS-MnasNetSS-E achieves a Top-1 accuracy of 75.7% on the ILSVRC2012 dataset, which surpasses MnasNet-A1 by 0.5%.

EfficientNet Search Space (EfficientNetSS). To compare with the state-of-the-art architecture EfficientNet-B0 [56], we also use HourNAS to search on the same search space as EfficientNet, which contains 4×10^{18} architectures. Targeting at EfficientNet-B0, we use its model size and FLOPs as two objectives to regularize space proposals and name the searched architecture as HourNAS-EfficientNetSS-F. Same as EfficientNet[3], we use the Swish [47] activation and Exponential Moving Average (EMA) in fully training. Note that the AutoAugment [12] is not used. The result in Tab. 1 shows HourNAS-EfficientNetSS-F surpasses EfficientNet-B0 by 0.2% Top-1 accuracy with similar number of parameters and FLOPs.

4.3. Ablation Study

If we do not restrict the computational resources of the sampled architectures in searching, the most complex block achieves the highest probability after searching for enough time. As shown in Tab. 3, we train the most complex architectures in both FBNet and EfficientNet search spaces, namely FBNet-Max and EfficientNet-Max. These two models obtain 75.7%, and 78.3% Top-1 accuracies, respectively. However, the computational resource requirements of these structures are relatively high. Therefore, the neural archi-

[3]https://github.com/tensorflow/tpu/tree/master/models/official/efficientnet

Table 3. The results of FBNet-Max and EfficientNet-Max on ILSVRC2012 dataset.

Model	Params (M)	FLOPS (M)	Top-1 (%)	Top-5 (%)
FBNet-Max	5.7	583	75.7	92.8
EfficientNet-Max	5.8	738	78.3	94.0

Table 4. Comparisons of searching with and without vital block priori on ILSVRC2012 dataset. The search spaces are original (upper) and enlarged (lower) FBNet search space, respectively.

Model	Type	Search Dataset	Search Cost (GPU days)	Params (M)	FLOPS (M)	Top-1 (%)	Top-5 (%)
HourNAS-FBNetSS-A	auto	ImageNet	**0.1**	4.8	298	**74.1**	**91.8**
HourNAS-FBNetSS-G (w/o vital block priori)	auto	ImageNet	0.2	4.7	297	73.2	91.4
HourNAS-EFBNetSS-C	auto	ImageNet	**0.1**	4.8	296	**74.1**	**91.6**
HourNAS-EFBNetSS-H (w/o vital block priori)	auto	ImageNet	0.2	4.8	299	73.5	91.3

Table 5. The results comparison on ILSVRC2012 dataset.

Model	Type	Search Dataset	Search Cost (GPU days)	Params (M)	FLOPS (M)	Top-1 (%)	Top-5 (%)
HourNAS-FBNetSS-A	auto	ImageNet	0.1	4.8	298	74.1	91.8
HourNAS-FBNetSS-I	auto	ImageNet	1.0	4.8	318	74.2	91.8

tecture search (NAS) could be regarded as the problem of computational resource allocation given the resource constraints.

The Impact of Vital Block Priori. In order to investigate the impact of the vital block priori, we directly search architectures without using the vital block information. All the blocks in the SuperNet S are treated equally in searching. We use the Gumbel-Max sampling and space proposal strategy to search architectures under the same predefined computational resources.

We use the previously described original and enlarged FBNet search spaces. We optimize 8 space proposals and it takes 6 hours for searching, which is twice of the counterpart that utilize the vital block priori. As shown in Tab. 4, the Top-1 accuracies of the discovered models (HourNAS-FBNetSS-G, HourNAS-EFBNetSS-H) drop by 0.9% and 0.6% on the ImageNet validation set, respectively. The searched vital blocks of HourNAS-FBNetSS-A uses 0.9M parameters and 130M FLOPs, and the HourNAS-FBNetSS-G uses 0.5M parameters and 55M FLOPs. The vital blocks in HourNAS-FBNetSS-G are not as expressive as HourNAS-FBNetSS-A, which results in worse performance. The results show the necessity of the vital block priori. Searching the vital blocks with higher priority is helpful in finding high-quality architectures. Therefore, we use a two-stage search method to allocate resources to vital blocks first, which can allocate resources more effectively, so as to complete the architecture search in a short time. The architectures are provided in the supplementary file, under same computational resources constraints, inclining more resources on the vital blocks gains more performance profit.

The Impact of Gumbel-Max Sampling. As discussed in Sec. 3.3, there are several design choices for the sampling methods. To find out the impact of the Gumbel-Max sampling method, here we instead use the Gumbel softmax [62] to optimize architecture parameters and network parameters. The search space and target resource constraints are the same as HourNAS-FBNetSS-A. The search process takes around 1 GPU day and the finalized architecture is denoted as HourNAS-FBNetSS-I. As shown in Tab. 5, HourNAS-FBNetSS-I outperforms HourNAS-FBNetSS-A by 0.1% Top-1 accuracy with much less searching cost, which demonstrate that Gumbel-Max is an efficient strategy for optimizing the SuperNet with almost no less of accuracy.

5. Conclusions

This paper investigates an efficient algorithm to search deep neural architectures on the large-scale dataset (*i.e.*, ImageNet) directly. To reduce the complexity of the huge search space, we present an Hourglass-based search framework, namely HourNAS. The entire search space is divided into "vital" and "non-vital" parts accordingly. By gradually search the components in each part, the search cost can be reduced significantly. Since the "vital" parts are more important for the performance of the obtained neural network, the optimization on this part can ensure accuracy. By exploiting the proposed approach, we can directly search architectures on the ImageNet dataset that achieves a 77.0% Top-1 accuracy using only 3 hours (*i.e.*, about 0.1 GPU days), which outperforms the state-of-the-art methods in both terms of search speed and accuracy.

Acknowledgement This work is supported by National Natural Science Foundation of China under Grant No. 61876007, and Australian Research Council under Project DE180101438 and DP210101859.

References

[1] Gabriel Bender, Hanxiao Liu, Bo Chen, Grace Chu, Shuyang Cheng, Pieter-Jan Kindermans, and Quoc V Le. Can weight sharing outperform random architecture search? an investigation with tunas. *CVPR*. 2

[2] Ermao Cai, Da-Cheng Juan, Dimitrios Stamoulis, and Diana Marculescu. Neuralpower: Predict and deploy energy-efficient convolutional neural networks. *ACML*, 2017. 1

[3] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once for all: Train one network and specialize it for efficient deployment. *ICLR*, 2020. 1

[4] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *ICLR*, 2019. 3, 7

[5] Jianlong Chang, xinbang zhang, Yiwen Guo, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Data: Differentiable architecture approximation. *NeurIPS*, 2019. 4, 6

[6] Hanting Chen, Yunhe Wang, Chunjing Xu, Boxin Shi, Chao Xu, Qi Tian, and Chang Xu. Addernet: Do we really need multiplications in deep learning? *CVPR*, 2020. 1

[7] Hanting Chen, Yunhe Wang, Chang Xu, Zhaohui Yang, Chuanjian Liu, Boxin Shi, Chunjing Xu, Chao Xu, and Qi Tian. Data-free learning of student networks. *ICCV*, 2019. 1

[8] Hanlin Chen, Li'an Zhuo, Baochang Zhang, Xiawu Zheng, Jianzhuang Liu, David S. Doermann, and Rongrong Ji. Binarized neural architecture search. *AAAI*, 2020. 2

[9] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. *ICCV*, 2019. 6, 7

[10] Zhuo Chen, Jiyuan Zhang, Ruizhou Ding, and Diana Marculescu. Vip: Virtual pooling for accelerating cnn-based image classification and object detection. *WACV*, 2020. 1

[11] Ting-Wu Chin, Ruizhou Ding, Cha Zhang, and Diana Marculescu. Towards efficient model compression via learned global ranking. *CVPR*, 2020. 1

[12] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. *CVPR*, 2019. 7

[13] Jiequan Cui, Pengguang Chen, Ruiyu Li, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast and practical neural architecture search. *ICCV*, 2019. 7

[14] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv*, 2017. 6

[15] Xiaohan Ding, guiguang ding, Xiangxin Zhou, Yuchen Guo, Jungong Han, and Ji Liu. Global sparse momentum sgd for pruning very deep neural networks. *NeurIPS*, 2019. 2

[16] Xuanyi Dong and Yi Yang. One-shot neural architecture search via self-evaluated template network. *ICCV*, 2019. 2

[17] Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. *CVPR*, 2019. 4, 6, 7

[18] Xuanyi Dong and Yi Yang. Nas-bench-102: Extending the scope of reproducible neural architecture search. *ICLR*, 2020. 1

[19] Thomas Elsken, Jan Metzen, and Frank Hutter. Efficient multi-objective neural architecture search via lamarckian evolution. *ICLR*, 2019. 4

[20] Ross Girshick. Fast r-cnn. *ICCV*, 2015. 1

[21] Xinyu Gong, Shiyu Chang, Yifan Jiang, and Zhangyang Wang. Autogan: Neural architecture search for generative adversarial networks. *ICCV*, 2019. 2

[22] Jianyuan Guo, Kai Han, Yunhe Wang, Chao Zhang, Zhaohui Yang, Han Wu, Xinghao Chen, and Chang Xu. Hit-detector: Hierarchical trinity architecture search for object detection. *CVPR*, 2020. 2

[23] 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*, 2019. 1, 2, 7

[24] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. *CVPR*, 2020. 1

[25] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. *ICCV*, 2017. 1

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, 2016. 1, 3, 4, 7

[27] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. *ECCV*, 2018. 1

[28] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv*, 2017. 7

[29] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. *CVPR*, 2017. 1

[30] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. *ECCV*. 6

[31] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 6

[32] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, 2009. 1

[33] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. *UAI*. 1

[34] Mingbao Lin, Rongrong Ji, Yuxin Zhang, Baochang Zhang, Yongjian Wu, and Yonghong Tian. Channel pruning via automatic structure search. *IJCAI*, 2020. 2

[35] Shaohui Lin, Rongrong Ji, Chao Chen, Dacheng Tao, and Jiebo Luo. Holistic cnn compression via low-rank decomposition with knowledge transfer. *TPAMI*, 2019. 1

[36] Shaohui Lin, Rongrong Ji, Yuchao Li, Yongjian Wu, Feiyue Huang, and Baochang Zhang. Accelerating convolutional networks via global & dynamic filter pruning. *IJCAI*, 2018. 1

[37] Benlin Liu, Yongming Rao, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Metadistiller: Network self-boosting via meta-learned top-down distillation. *ECCV*, 2020. 1

[38] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. *CVPR*, 2019. 2

[39] 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. *ECCV*, 2018. 2, 7

[40] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *ICLR*, 2019. 1, 2, 4, 7

[41] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. *ECCV*, 2016. 1

[42] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh D. Dhebar, Kalyanmoy Deb, Erik D. Goodman, and Wolfgang Banzhaf. Nsga-net: A multi-objective genetic algorithm for neural architecture search. *GECC*, 2018. 2, 4

[43] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. *ECCV*, 2018. 7

[44] Chris J Maddison, Daniel Tarlow, and Tom Minka. A* sampling. *NeurIPS*, 2014. 6

[45] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *ICML*, 2018. 2

[46] Ruijie Quan, Xuanyi Dong, Yu Wu, Linchao Zhu, and Yi Yang. Auto-reid: Searching for a part-aware convnet for person re-identification. *ICCV*, 2019. 2

[47] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv*, 2017. 7

[48] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. *AAAI*, 2019. 1, 2, 7

[49] Mark B. Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *CVPR*, 2018. 3, 7

[50] Albert Shaw, Wei Wei, Weiyang Liu, Le Song, and Bo Dai. Meta architecture search. *NeurIPS*, 2019. 1

[51] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015. 1

[52] Dimitrios Stamoulis, Ting-Wu Rudy Chin, Anand Krishnan Prakash, Haocheng Fang, Sribhuvan Sajja, Mitchell Bognar, and Diana Marculescu. Designing adaptive neural networks for energy-constrained image classification. *ICCAD*, 2018. 1

[53] Dimitrios Stamoulis, Ruizhou Ding, Di Wang, Dimitrios Lymberopoulos, Bodhi Priyantha, Jie Liu, and Diana Marculescu. Single-path nas: Designing hardware-efficient convnets in less than 4 hours. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2019. 3

[54] Dimitrios Stamoulis, Ruizhou Ding, Di Wang, Dimitrios Lymberopoulos, Bodhi Priyantha, Jie Liu, and Diana Marculescu. Single-path mobile automl: Efficient convnet design and nas hyperparameter optimization. *IEEE Journal of Selected Topics in Signal Processing*, 2020. 3

[55] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. *CVPR*, 2018. 2, 3, 4, 5, 6, 7

[56] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *ICML*, 2019. 3, 4, 5, 7

[57] Mingxing Tan and Quoc V. Le. Mixconv: Mixed depthwise convolutional kernels. *BMVC*, 2019. 6

[58] Yehui Tang, Yunhe Wang, Yixing Xu, Hanting Chen, Boxin Shi, Chao Xu, Chunjing Xu, Qi Tian, and Chang Xu. A semi-supervised assessor of neural architectures. *CVPR*, 2020. 1

[59] Yehui Tang, Yunhe Wang, Yixing Xu, Dacheng Tao, Chunjing Xu, Chao Xu, and Chang Xu. Scop: Scientific control for reliable neural network pruning. *arXiv*, 2020. 1

[60] Andreas Veit, Michael Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. *NeurIPS*, 2016. 2, 3

[61] Yunhe Wang, Yixing Xu, and Dacheng Tao. Dc-nas: Divide-and-conquer neural architecture search. *arXiv preprint arXiv:2005.14456*, 2020. 1

[62] 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. *CVPR*, 2019. 1, 2, 3, 4, 5, 6, 7, 8

[63] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. *ICLR*, 2019. 1, 2, 7

[64] Yunyang Xiong, Ronak Mehta, and Vikas Singh. Resource constrained neural network architecture search. *ICCV*, 2019. 1, 7

[65] Antoine Yang, Pedro M Esperança, and Fabio M Carlucci. Nas evaluation is frustratingly hard. *ICLR*, 2020. 1

[66] Zhaohui Yang, Yunhe Wang, Xinghao Chen, Boxin Shi, Chao Xu, Chunjing Xu, Qi Tian, and Chang Xu. Cars: Continuous evolution for efficient neural architecture search. *CVPR*, 2020. 4, 7

[67] Zhaohui Yang, Yunhe Wang, Chang Xu, Peng Du, Chao Xu, Chunjing Xu, and Qi Tian. Discernible image compression. *ACMMM*, 2020. 1

[68] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. *ICML*. 1

[69] Fuxun Yu, Zhuwei Qin, Di Wang, Ping Xu, Chenchen Liu, Zhi Tian, and Xiang Chen. Dc-cnn: computational flow redefinition for efficient cnn through structural decoupling. *Proceedings of the 23rd Conference on Design, Automation and Test in Europe*, 2020. 1

[70] Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. *ICLR*, 2019. 1

[71] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. *ICLR*, 2020. 1

[72] Chiyuan Zhang, Samy Bengio, and Yoram Singer. Are all layers created equal. *ICMLW*, 2019. 2

[73] Xiawu Zheng, Rongrong Ji, Lang Tang, Baochang Zhang, Jianzhuang Liu, and Qi Tian. Multinomial distribution learning for effective neural architecture search. *ICCV*, 2019. 7

[74] Li'an Zhuo, Baochang Zhang, Hanlin Chen, Linlin Yang, Chen Chen, Yanjun Zhu, and David S. Doermann. Cp-nas: Child-parent neural architecture search for 1-bit cnns. *IJCAI*, 2020. 2

[75] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *ICLR*, 2017. 1, 2

[76] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. *CVPR*, 2018. 1, 2, 7