LOCALLY FREE WEIGHT SHARING FOR NETWORK WIDTH SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Searching for network width is an effective way to slim deep neural networks with hardware budgets. With this aim, a one-shot supernet is usually leveraged as a performance evaluator to rank the performance w.r.t. different width. Nevertheless, current methods mainly follow a manually fixed weight sharing pattern, which is limited to distinguish the performance gap of different width. In this paper, to better evaluate each width, we propose a locally free weight sharing strategy (CafeNet) accordingly. In CafeNet, weights are more freely shared, and each width is jointly indicated by its base channels and free channels, where free channels are supposed to locate freely in a local zone to better represent each width. Besides, we propose to further reduce the search space by leveraging our introduced FLOPs-sensitive bins. As a result, our CafeNet can be trained stochastically and get optimized within a min-min strategy. Extensive experiments on ImageNet, CIFAR-10, CelebA and MS COCO dataset have verified our superiority comparing to other state-of-the-art baselines. For example, our method can further boost the benchmark NAS network EfficientNet-B0 by 0.41% via searching its width more delicately.

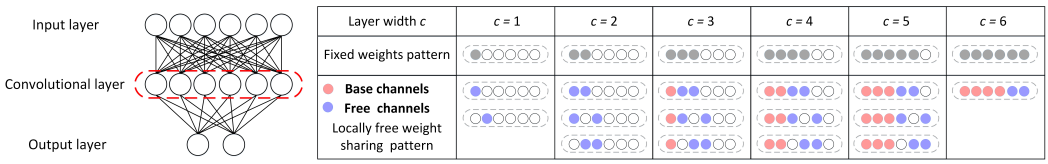
1 INTRODUCTION

Deep neural networks are easily constructed by stacking multiple layers of non-linearities on top of each other. But it is a much more delicate job than it seems. Before going into the deep, we have to determine the width of each layer[1] (Chen et al., 2020) to be stacked, especially given the consideration of the computational budgets (He et al., 2016; Han et al., 2015; Ioffe & Szegedy, 2015; Huang et al., 2017) (*e.g.*, FLOPs, and latency) and the resulting network performance. The width of a pretrained over-parameterized neural network can be further trimmed by classical channel pruning techniques (Liu et al., 2019b; He et al., 2017; Lin et al., 2018; Huang & Wang, 2018; Liu et al., 2017) to eliminate the redundancy. Inspired by the spark of neural architecture search (NAS) (Liu et al.; Cai et al.; Guo et al., 2019; Cai et al., 2019; Hu et al., 2020), recent methods attempt to directly search for the optimal network width for a given network, and have achieved remarkable performance in various FLOPs levels or acceleration rates, such as AutoSlim (Yu & Huang, 2019), MetaPruning (Liu et al., 2019a) and TAS (Dong & Yang, 2019).

To determine the optimal network width, we need to evaluate and compare the performance of different network width. One of the most straightforward yet authentic approaches is to examine their training-from-scratch performance and traverse all possible settings of network width (not exceeding a maximum allowable width). However, exhaustive training is computationally unaffordable due to the huge search space. Taking MobileNetV2 as an example, 100 channels in each of the 25 layers result in 100^{25} possible network architectures. For sake of searching efficiency, current methods Chen et al. (2020); Yu & Huang (2019); Dong & Yang (2019) usually follow a weight sharing strategy by leveraging a one-shot supernet, with various sub-networks sharing the same weights with the supernet. Then each network width can be efficiently evaluated by querying the performance of its corresponding sub-network in the supernet.

In this way, how to specify the sub-network for each network width indicates the search space and matters for the performance evaluation. Without loss of generality, we count channels in a layer from the left. A popular weight sharing strategy for the supernet then follows a fixed weight sharing

[1] Other literature also use the number of channels/filters to indicate the network width.



(a) Toy example (b) Channel assignment of sub-network w.r.t. differernt weight sharing patterns

Figure 1: (a) A toy example of the supernet. We only search for the width for its one convolutional layer with all 6 channels. (b) Examples of different weight sharing patterns. For the fixed weight sharing pattern, leftmost c channels are assigned for the sub-network of width c. For the locally free weight sharing pattern, the offset of free zone is set to $r = 1$.

pattern, which simply assigns the left c channels as the sub-network for the width c (see the gray dots in Fig.1(b)). In this manner, the search space of a layer with n channels is $\mathcal{O}(n)$. However, this fixed pattern imposes an inherent constraint on the search space. Any two sets of width fully share their weights, limiting the accurate evaluation ability of supernet and affecting the searching performance accordingly. Ideally, each channel from the supernet shall enjoy the freedom to be selected, *i.e.*, any channel has a probability to occur in the subnetwork, no matter it locates at the leftmost or the rightmost of a layer. But the price of this *full* channel freedom is the dramatic increase of the search space size to $\mathcal{O}(2^n)$, which affects the feasibility of width search. Instead of this unaffordable full freedom, we prefer restrictions and regularization over the search space to keep the channel freedom within an acceptable range, which benefits the efficiency of width search.

In this paper, we propose a locally free weight sharing strategy of the supernet (*i.e.*, CafeNet) for network width search. For each width c, our goal is to specify a sub-network within supernet by assigning c channels, indicated by an index set $\mathcal{I}(c)$. To balance the weight sharing freedom and the searching efficiency, we split the index set $\mathcal{I}(c)$ into two parts, *i.e.*, c_b *base channels* $\mathcal{I}_b(c)$ and c_f *free channels* $\mathcal{I}_f(c)$, with $\mathcal{I}(c) = \mathcal{I}_b(c) \cup \mathcal{I}_f(c)$ and $c = c_b + c_f$. (see Fig.1(b)). In detail, base channels follow the fixed weight sharing pattern, while free channels are encouraged to select freely from a given zone, which is the neighborhood of the c-th channel. In this way, each width c can be better evaluated by multiple locally free sub-subnetworks. The supernet CafeNet can be trained by optimizing the sub-network with the minimum loss for each width and search for the one with the maximum accuracy. To further ease the burden of searching cost, we take a group of channels (*i.e.*, *bin*) as the minimal searching unit. The bin sizes are carefully set by encouraging FLOPs to be more evenly allocated over bins. We conduct extensive experiments on the benchmark ImageNet (Deng et al., 2009) and CIFAR-10 (Krizhevsky et al., 2014) datasets to validate the superiority of our proposed CafeNet. The effectiveness of our CafeNet is also examined on the face attribute recognition task on CelebA (Liu et al., 2015) dataset and the transferability of our searched backbone for object detection task on MS COCO dataset (Everingham et al., 2010).

2 Problem Formulation

With a given network structure, the aim of the network width search is to optimize its width for each layer under certain budgets. Suppose the target network has L layers, then the network width can be represented as a L-length tuple of layer-wise width $\boldsymbol{c} = (c_1, c_2, \ldots, c_L)$ in which c_i represents the width at the i-th layer. Without loss of clarity, we simply focus on searching for the optimal layer width of c_i. Denote the maximum allowable width at the i-th layer as n_i. Then the number of all sub-networks in the supernet amounts to $\prod_{i=1}^{L}(2^{n_i} - 1)$, which is absolutely huge. Current methods mainly use weight sharing strategy to boost the searching efficiency. In the typically fixed weight sharing pattern, each layer width $c_i \in [1 : n_i] = \{1, 2, ..., n_i\}$ is indicated by one and only one sub-network (see Fig.1(b)), and all layers within the sub-network adopt the same weight sharing pattern, then the size of search space $\mathcal{C}$ is reduced to $\prod_{i=1}^{L} n_i$.

After specifying each width with its corresponding sub-network, we can evaluate it by examining the performance of its sub-network in the supernet with shared weights. Then the whole searching consists of two steps, *i.e.*, supernet training and searching with supernet. Usually, the original training dataset is split into two datasets, *i.e.*, training dataset $\mathcal{D}_{tr}$ and validation dataset $\mathcal{D}_{val}$. The one-shot supernet $\mathcal{N}$ with weights W is trained by randomly sampling a width $\boldsymbol{c}$ and optimizing its

corresponding sub-network with weights $\boldsymbol{w_c} \subset W$, *i.e.*,

$$W^* = \arg\min_{\boldsymbol{w_c} \subset W} \mathop{\mathbb{E}}_{\boldsymbol{c} \in U(\mathcal{C})} [\![\mathcal{L}_{train}(\boldsymbol{w}_c; \mathcal{N}, \boldsymbol{c}, \mathcal{D}_{tr})]\!], \tag{1}$$

where $U(\mathcal{C})$ is a uniform distribution of network width in search space, and $\mathbb{E}[\![\cdot]\!]$ is the expectation of random variables. After the supernet is trained, the performance (*e.g.*, classification accuracy) of various network width is indicated by evaluating its sub-network on the trained supernet $\mathcal{N}^*$. And the optimal width corresponds to that of the highest performance on the validation dataset, *e.g.*,

$$\boldsymbol{c}^* = \arg\max_{\boldsymbol{c} \in \mathcal{C}} \text{ Accuracy}(\boldsymbol{w}^*_{\boldsymbol{c}}, W^*; \mathcal{N}^*, \mathcal{D}_{val}), \text{ s.t. FLOPs}(\boldsymbol{c}) \leq F_b, \tag{2}$$

where F_b is a constraint on the FLOPs, here we consider FLOPs rather than latency as the hardware constraint since we are not targeting any specific hardware device like EfficientNet Tan & Le (2019) and other channel pruning baselines Zhuang et al. (2018); Ye et al. (2020); Dong & Yang (2019); He et al. (2018b). The searching of Eq.(2) can be fulfilled efficiently by various algorithms, such as random or evolutionary search (Hancock, 1992; Whitley et al., 1990; Liu et al., 2019a).

3 CAFENET

3.1 LOCALLY FREE WEIGHT SHARING

As illustrated before, a one-shot supernet $\mathcal{N}$ is usually trained as an evaluator to reflect the performance of different settings of network width. Assume we count channels in a layer from the left. To locate the sub-network in the supernet for width c at a certain layer [2], conventional fixed weight sharing pattern assigns the leftmost c channels as the sub-network as Fig.1(b). Formally, for width c, its corresponding sub-network is specified by a fixed *channel assignment* with an index set $\mathcal{I}(c)$ of channels, *i.e.*,

$$\text{fixed weight sharing pattern:} \quad \mathcal{I}(c) = [1 : c]. \tag{3}$$

However, this hard assignment imposes an inherent constraint on the search space. Channel assignment of smaller width will always be a subset of that of larger width. Actually, this fixed pattern forces the weight sharing of two different width to the greatest extent. To quantify this property, we define a simple weight sharing degree d of two any width c and $\tilde{c}$ by examining the overlap of their channel assignment, *i.e.*,

$$d(c, \tilde{c}) = \frac{|\mathcal{I}(c) \cap \mathcal{I}(\tilde{c})|}{\min(|\mathcal{I}(c)|, |\mathcal{I}(\tilde{c})|)} \in [0, 1]. \tag{4}$$

Intuitively, if the degree $d(c, \tilde{c})$ is larger, then sub-networks of width c and $\tilde{c}$ will share more weights, and this implies that the supernet will enforce width c and $\tilde{c}$ to have more similar performance. And for the fixed pattern, the weight sharing degree reaches its maximum as Theorem 1 [3].

Theorem 1. *The weight sharing degree of fixed pattern will always be 1 for any two width.*

However, ideally, each width should have full freedom in the supernet to select its channel assignment. In this situation, sub-networks of two different layer width can be fully shared, partially shared, or even have no overlap, thus the weight sharing degree freely varies from 0 to 1. And each width can be better evaluated by some sub-networks, instead of a manually assigned sub-network. Nevertheless, this induces the search space to proliferate from the fixed $\mathcal{O}(n)$ to $\mathcal{O}(2^n)$, which is computationally unaffordable for practical search.

Therefore, we bridge these two extreme situations, and propose a locally free weight sharing strategy. Basically, to specify a sub-network of width c from a supernet for evaluation, we need to assign c channels from all n channels per layer, with the channel assignment index set $\mathcal{I}(c)$. We split the index set $\mathcal{I}(c)$ into two parts, *i.e.*, c_b base channels $\mathcal{I}_b(c)$ and c_f free channels $\mathcal{I}_f(c)$, with $\mathcal{I}(c) = \mathcal{I}_b(c) \cup \mathcal{I}_f(c)$ and $c = c_b + c_f$. In detail, for free channels, we encourage to select c_f channels from a given zone, which is the neighborhood of the c-th channel, *i.e.*, $\mathcal{B}(c; r) = [c - r : c + r]$, and r is a preset allowed offset to control the search space with $c_f = r + 1$. Besides, base channels follow the fixed weight sharing pattern with $\mathcal{I}_b(c) = [0 : c_b]$; let $\mathcal{I}_b(c) \cap \mathcal{I}_f(c) = \emptyset$, then we can have

[2] Here we omit the subscript i for indicating any width c_i at i-th layer.

[3] It holds naturally. Suppose $c \leq \tilde{c}$, then we have $\mathcal{I}(c) \cap \mathcal{I}(\tilde{c}) = \mathcal{I}(c)$ for fixed pattern.

$c_b = \max(c - r - 1, 0)$. In this way, in our formulation, we can specify multiple channel assignment for width c to better evaluate its performance as Fig.1(b), *i.e.*,

$$\text{locally free weight sharing pattern:} \quad \mathcal{I}(c) = [1 : c_b] \cup \mathcal{I}_f(c), \text{ where } \mathcal{I}_f(c) \subset [c - r : c + r] \quad (5)$$

According to Eq.(4), the weight sharing degree of our method is between $c_b/c \leq d(c, \tilde{c}) \leq 1$. By properly setting offset r, we can achieve a tradeoff between the fixed pattern and the fully free pattern. Given a r, now our search space scales at $\mathcal{O}(\mathbb{C}_{2r+1}^{r+1} n)$, which is only a constant times (*e.g.*, $\mathcal{O}(3n)$ for $r = 1$, see Fig.1(b)) of fixed weight sharing search space. In this way, each width can be better represented, thus boosts the performance ranking ability of supernet.

Training with min-min optimization. With the above search space, we can train our CafeNet in a stochastic setting as Eq.(1), which simply samples a network width $\boldsymbol{c}$ first, and then optimizes its sub-network. Nevertheless, in CafeNet a width is specified more freely by several sub-networks, and the performance of these sub-networks can be different. In this way, we propose to indicate the performance for each width by examining its sub-network with the best performance. In training CafeNet, instead of randomly optimizing a sub-network, we optimize the sub-network with the smallest training loss. To achieve this, we need to traverse all sub-networks for width $\boldsymbol{c}$. However, this can be efficiently fulfilled since our job is to identify a sub-network with the minimum loss, which can be quickly calculated in a feed-forward manner. Afterward, we only need to implement a backward update for the target sub-network with minimum loss. A detailed explanation of this min-min optimization is elaborated in Appendix A.1.

Searching with max-max selection. After the CafeNet $\mathcal{N}$ is trained, we can evaluate each network width by examining its performance (*e.g.* classification accuracy) on the validation dataset $\mathcal{D}_{val}$. Similar to the training of CafeNet, for each width $\boldsymbol{c}$, we use the sub-network with the highest performance to indicate the performance. Then we search for the optimal $\boldsymbol{c}^*$ as Eq.(2). Note that since searching itself is much faster than the supernet training, the increased computational cost is subtle and acceptable in real practice. A detailed description of this max-max selection is presented in Appendix A.1.

3.2 Reducing search space with FLOPs-sensitive bins

Based on the locally free weight sharing pattern, each network width can be more flexibly indicated and then evaluated by the CafeNet. However, as previously illustrated, the size of the search space is considerably large and brings a tough challenge for the subsequent searching algorithms, such as random search and evolutionary search. In this way, current methods usually choose to reduce the search space by initializing the *minimum searching unit* as a group of channels (called *bin*) instead of a single channel. Concretely, all channels at each layer are partitioned into K groups, with (n_i/K) channels in each group for the i-th layer.

However, this uniform partition neglects the extra information of channels within layers, *i.e.*, kernel size and feature map resolution, and all of these factors are involved in the calculation of FLOPs. This inspires us that we should build bins based on the FLOPs. For simplicity, we encourage all bins in our search space to have the same FLOPs. Note that FLOPs is up to the layer width between two adjacent layers; for example, for convolution with resolution $H \times W$ and kernel size $K \times K$, its FLOPs can be calculated as

$$\text{FLOPs}(c_{in}, c_{out}) = c_{in} \times c_{out} \times H \times W \times K^2, \quad (6)$$

where c_{in} and c_{out} are the width for the input and output of this layer, respectively. In this way, channels at different layers contribute differently to the overall FLOPs of the supernet. And if a channel at a layer contributes more to the FLOPs, then we can say it is more sensitive w.r.t. FLOPs. This *FLOPs-sensitivity* can be simply reflected by examining the real FLOPs of a single channel at a layer. Concretely, for each channel at the i-th layer, it serves as both the output channel of the $(i-1)$-th layer and input channel of the $(i+1)$-th layer. Thus the sensitivity ε_i for the i-th layer can be represented as

$$\varepsilon_i = \text{FLOPs}(n_{i-1}, 1) + \text{FLOPs}(1, n_{i+1}), \quad (7)$$

where n_{i-1} and n_{i+1} are the width of the $(i-1)$-th and the $(i+1)$-th layer, respectively.

As a result, we simply encourage all bins to have the same FLOPs-sensitivity. To do so, we propose to initialize the bin size (*i.e.*, number of channels in a bin) more adaptively. The intuition is that the

larger FLOPs sensitivity a layer has, the fewer channels should be allocated in a bin, and vice versa. As a result, the bin size b_i for i-th layer should satisfy

$$b_i \propto 1/\varepsilon_i. \tag{8}$$

In practice, we usually specify a minimum bin size (*i.e.*, with maximum sensitivity ε) as β, then the bin size for each layer is

$$b_i = \beta \times \frac{\max_j \varepsilon_j}{\varepsilon_i}. \tag{9}$$

Actually, the minimum bin size β in Eq.(9) controls the total number of bins, thus also determines the size of the search space. If β is set to be smaller, the search space will be much larger and fine-grained since our search unit gets smaller and delicate accordingly. We will discuss a multi-stage search strategy by controlling β actively in Section 3.3, and present empirical investigations in Appendix A.16.

3.3 Further boosting performance with multi-stage search

As previously illustrated, setting minimum bin size β in Eq.(9) to be smaller will bring in a smaller searching unit, and thus a more fine-grained structure will be expected. However, this is accompanied by a larger search space and is challenging for both the ranking ability of supernet and searching performance. In this way, we can leverage a multi-stage searching strategy to avoid the search space being too large but still can search on a fine-grained level. In detail, we propose to search for the width under the multi-stage FLOPs budget, which is decayed linearly, *i.e.*,

$$\text{FLOPs}(t) \leftarrow \text{FLOPs}(0) - (\text{FLOPs}(0) - F_b) \times \frac{t}{T}, \quad \beta(t+1) \leftarrow \beta(t)/\alpha \tag{10}$$

where $\text{FLOPs}(t)$ represents the FLOPs budget in t-th stage and $\text{FLOPs}(0)$ is the FLOPs of the supernet. F_b is the target FLOPs after all T stages as Eq.(13). Besides, at each stage, we shrink the minimum bin size β with evolving speed $\alpha > 0$. Thus the search space will evolve from the coarse-grained to the fine-grained, but with controllable size. More details are explored in Appendix A.14 and A.15. However, searching with the multi-stage strategy inevitably introduces more computation consumption. Therefore, we only implement this method in the small dataset (*i.e.* CIFAR-10).

4 Experimental Results

In this section, we perform extensive experiments to validate the effectiveness of our CafeNet. For all architectures, we use the SGD optimizer with momentum 0.9. Parameters β in Eq.(9) and α in Eq.(10) are set to 1 and 2, respectively. Offset r for locally free channels is set to 1 for all experiments. Detailed experimental settings are elaborated in Appendix A.2.

We include various state-of-the-art width search methods for comparison, *e.g.* AutoSlim (Yu & Huang, 2019), MetaPruning (Liu et al., 2019a), TAS (Dong & Yang, 2019), GBN (You et al., 2019) and FPGM (He et al., 2019a). Besides, for comprehensive comparison, we perform searching using both evolutionary search and random search, named as **CafeNet-E** and **CafeNet-R**, respectively. In addition, we also consider two vanilla baselines, *i.e.*, Uniform and Random. Uniform: we shrink the width of each layer uniformly with a fixed factor to meet the FLOPs budget. Random: we randomly sample 20 networks under FLOPs constraint, and train them by 50 epochs, then we continue training the one with the highest performance and report its final result.

4.1 Results on ImageNet Dataset

We perform experiments on ResNet50 (He et al., 2016), MobileNetV2 (Sandler et al., 2018) and EfficientNet-B0 (Tan & Le, 2019) to examine the performance of CafeNet on heavy and light models, as summarized in Table 1 and 2. In detail, ResNet50 and MobileNetV2 are searched from the original size, and for EfficientNet-B0, the 0.5× FLOPs is also searched from the original model. But the 1.0× FLOPs is from a supernet with 1.5× FLOPs from uniform width scaling. The original EfficientNet-B0, ResNet50 and MobileNetV2 has 5.3M, 25.5M and 3.5M parameters and 386M, 4.1G and 300M FLOPs with 76.4%, 77.8% and 73.7% Top-1 accuracy, respectively. More detailed experiments and visualization results of our searched network are shown in Appendix A.8 and A.11.

Table 1: Results on ImageNet of ResNet50 and MobileNetV2 compared with state-of-the-art methods. References of baseline methods are summarized in the Appendix A.4.

ResNet50						MobileNetV2					
	Methods	FLOPs	Param	Top-1	Top-5		Methods	FLOPs	Param	Top-1	Top-5
3G	AutoSlim	3.0G	23.1M	76.0%	-	200M	MetaPruning	217M	-	71.2%	-
	MetaPruning	3.0G	-	76.2%	-		LEGR	210M	-	71.4%	-
	LEGR	3.0G	-	76.2%	-		AutoSlim	207M	4.1M	73.0%	-
	Uniform	3.0G	19.1M	75.9%	93.0%		Uniform	217M	2.7M	70.9%	89.4%
	Random	3.0G	-	75.2%	92.5%		Random	217M	-	70.3%	89.1%
	CafeNet-R	3.0G	22.6M	**77.1%**	94.3%		**CafeNet-R**	217M	3.0M	**73.3%**	91.1%
	CafeNet-E	3.0G	23.8M	**77.4%**	94.5%		**CafeNet-E**	217M	3.3M	**73.4%**	91.2%
2G	GBN	2.4G	31.8M	76.2%	92.8%	150M	LEGR	180M	-	70.8%	-
	LEGR	2.4G	-	75.7%	92.7%		TAS	150M	-	**70.9%**	-
	FPGM	2.4G	-	75.6%	92.6%		AMC	150M	-	**70.8%**	-
	TAS	2.3G	-	76.2%	93.1%		LEGR	150M	-	69.4%	-
	AutoSlim	2.0G	20.6M	75.6%	-		MuffNet	149M	-	63.7%	-
	Uniform	2.0G	13.3M	75.1%	92.7%		Uniform	150M	2.0M	69.3%	88.9%
	Random	2.0G	-	74.6%	92.2%		Random	150M	-	68.8%	88.7%
	CafeNet-R	2.0G	19.1M	**76.5%**	93.1%		**CafeNet-R**	150M	2.7M	**71.9%**	90.0%
	CafeNet-E	2.0G	18.4M	**76.9%**	93.3%		**CafeNet-E**	150M	3.0M	**72.4%**	90.4%
	MetaPruning	1.0G	-	73.4%	-	100M	MetaPruning	105M	-	65.0%	-
	AutoSlim	1.0G	-	74.0%	-		Uniform	105M	1.5M	65.1%	89.6%
1G	Uniform	1.0G	6.6M	73.1%	91.8%		Random	105M	-	63.9%	-
	Random	1.0G	-	72.2%	91.4%		**CafeNet-R**	106M	2.2M	**68.2%**	88.2%
	CafeNet-R	1.0G	11.2M	**74.9%**	92.3%		**CafeNet-E**	106M	2.1M	**68.7%**	88.5%
	CafeNet-E	1.0G	12M	**75.3%**	92.6%		MuffNet	50M	-	50.3%	-
570M	AutoSlim	570M	-	72.2%	-		MetaPruning	43M	-	58.3%	-
	Uniform	570M	4.0M	71.6%	90.6%		Uniform	50M	0.9M	59.7%	82.0%
	Random	570M	-	69.4%	90.3%		Random	50M	-	57.4%	81.2%
	CafeNet-R	570M	11.3M	**72.7%**	90.9%		**CafeNet-R**	50M	1.7M	**64.3%**	85.2%
	CafeNet-E	570M	12.0M	**73.3%**	91.2%		**CafeNet-E**	50M	1.6M	**64.9%**	85.4%

Table 2: Searching results of EfficientNet-B0 with $1\times$ and $0.5\times$ FLOPs on ImageNet dataset.

$1\times$ EfficientNet-B0					$0.5\times$ EfficientNet-B0				
Methods	FLOPs	Param	Top-1	Top-5	Methods	FLOPs	Param	Top-1	Top-5
Uniform	385M	5.3M	76.42%	92.24%	Uniform	192M	2.8M	74.16%	91.48%
Random	385M	5.1M	75.96%	92.11%	Random	192M	3.0M	73.36%	91.03%
CafeNet-R	385M	6.2M	**76.59%**	92.74%	**CafeNet-R**	192M	3.7M	**74.47%**	91.65%
CafeNet-E	385M	6.9M	**76.83%**	92.96%	**CafeNet-E**	192M	3.9M	**74.62%**	91.87%

As shown in Table 1, our CafeNet achieves the highest performance on ResNet50 and MobileNetV2 w.r.t. different FLOPs, which proves the superiority of our method compared to other width search algorithms. Besides, CafeNet obtains remarkable performance with tiny FLOPs budgets. For example, our 570M ResNet50 achieves 73.3% Top-1 accuracy, surpassing AutoSlim (Yu & Huang, 2019) by 1.1%. In addition, as shown in Table 2, for the NAS-based network EfficientNet-B0, the performance can be further improved using CafeNet. It indicates that our CafeNet can optimize its width in a more fine-grained manner.

4.2 Results on CIFAR-10 Dataset

We also investigate CafeNet with MobileNetV2 (Sandler et al., 2018) and VGGNet (Simonyan & Zisserman, 2014) on the moderate CIFAR-10 dataset. Our original VGGNet (MobileNetV2) has 20M (2.2M) parameters and 399M (297M) FLOPs with accuracy of 93.99% (94.81%). As shown in Table 3, our CafeNet enjoys significant superiority to other algorithms. In detail, our 44M MobileNetV2 achieves 95.31% accuracy and outperforms MuffNet (Chen et al., 2019) by 2.2%. For VGGNet, our CafeNet achieves 94.36% accuracy with 189M FLOPs, which surpasses DCP (Zhuang et al., 2018) by 0.2%. Similarly, our CafeNet shows great advantages w.r.t. tiny models, *e.g.* with 2.11% accuracy improvement than AutoSlim (Yu & Huang, 2019) on MobileNetV2 with 28M FLOPs. More experiments on CIFAR-10 dataset are investigated in Appendix A.9.

4.3 Experiments on face attribute task with CelebA dataset

To futher investigate the effectiveness of CafeNet, we perform experiments of multi-label classification task on CelebA (Liu et al., 2015) dataset with MobileNetV2 (Sandler et al., 2018) and ResNet18 (He et al., 2016). CelebA dataset has 162K training and 20K testing images with 40 attribute labels from 2 categories. We report the average accuracy of all 40 labels. The original MobileNetV2 (ResNet18) has 213M (1.3G) FLOPs and 2.2M (11.2M) parameters with 92.07% (92.13%) average accuracy.

Table 3: Results of MobileNetV2 and VGGNet on CIFAR-10 dataset. References of baseline methods are summarized in Appendix A.4.

MobileNetV2					VGGNet				
Groups	Methods	FLOPs	Parameters	Accuracy	Groups	Methods	FLOPs	Parameters	Accuracy
100M+	DCP	218M	1.7M	94.75%	200M	GAL	190M	-	93.80%
	CafeNet-R	188M	1.4M	**95.44%**		DCP	199M	10.4M	94.16%
	CafeNet-E	188M	1.5M	**95.56%**		Sliming	199M	10.4M	93.80%
	MuffNet	175M	-	94.71%		**CafeNet-R**	189M	8.3M	**94.27%**
	CafeNet-R	144M	1.2M	**95.28%**		**CafeNet-E**	189M	8.0M	**94.36%**
	CafeNet-E	144M	1.1M	**95.44%**	100M+	PS	156M	-	93.63%
100M-	AutoSlim	88M	1.5M	93.20%		**CafeNet-R**	154M	3.4M	**94.09%**
	AutoSlim	59M	0.7M	93.00%		**CafeNet-E**	154M	3.1M	**94.23%**
	MuffNet	45M	-	93.12%		AOFP	124M	-	93.84%
	CafeNet-R	44M	0.4M	**95.16%**		**CafeNet-R**	115M	2.4M	**93.87%**
	CafeNet-E	44M	0.4M	**95.31%**		**CafeNet-E**	115M	2.1M	**94.01%**
	AutoSlim	28M	0.3M	92.00%	76M	CGNets	92M	-	92.88%
	CafeNet-R	28M	0.2M	**93.87%**		**CafeNet-R**	76M	2.2M	**93.36%**
	CafeNet-E	28M	0.2M	**94.11%**		**CafeNet-E**	76M	1.4M	**93.67%**

Table 4: Results of MobileNetV2 and ResNet18 on CelebA dataset.

MobileNetV2					ResNet18				
FLOPs	21M	51M	106M	162M	FLOPs	130M	316M	619M	1G
Uniform	91.63%	91.73%	91.92%	91.97%	Uniform	91.66%	91.79%	91.93%	92.03%
Random	91.42%	91.52%	91.63%	91.76%	Random	91.51%	91.62%	91.67%	91.84%
CafeNet-R	**91.71%**	**92.03%**	**92.09%**	**92.12%**	**CafeNet-R**	**91.83%**	**92.07%**	**92.13%**	**92.17%**
CafeNet-E	**91.85%**	**92.13%**	**92.16%**	**92.19%**	**CafeNet-E**	**91.92%**	**92.16%**	**92.18%**	**92.25%**

Table 5: Results of ResNet50 and MobileNetV2 on MS COCO dataset (Lin et al., 2017b;a).

ResNet50				MobileNetV2			
Framework	Original 1.0×	CafeNet 0.5×	Uni 0.5×	Framework	Original 1.0×	CafeNet 0.5×	Uni 0.5×
RetinaNet	36.3%	34.9%	34.2%	RetinaNet	31.2%	29.2%	27.5%
Faster R-CNN	37.2%	36.0%	35.5%	Faster R-CNN	31.7%	29.1%	28.5%

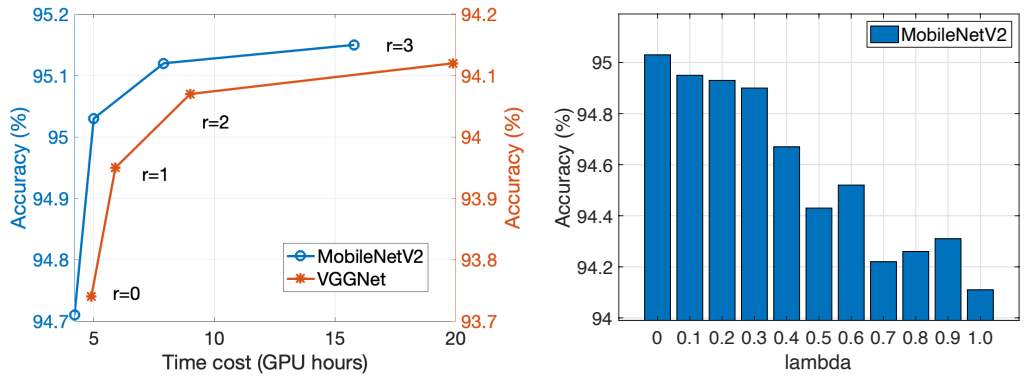
As shown in Table 4, our CafeNet can even achieve higher accuracy with FLOPs reducing a bit. For example, our 162M-FLOPs MobileNetV2 has 92.19% average accuracy and outperforms the original model by 0.12%. Besides, our algorithm still enjoys great superiority in tiny FLOPs; for example, our 130M-ResNet18 achieves 91.92% average accuracy, which is 0.26% higher than the uniform baseline. Besides, to comprehensively compare the classification performances on all 40 labels, we present the comparison results for each label in Appendix A.9.

4.4 Transferability of the searched width to object detection task

For object detection tasks, its backbone is usually initialized by a pretrained model on ImgeNet dataset. In this way, we use MobileNetV2 and ResNet50 with 0.5× FLOPs as the backbone feature extractors to examine the transferability of our searched network width. Besides, we leverage both the two-stage Faster R-CNN with Feature Pyramid Networks (FPN) (Lin et al., 2017a; Ren et al., 2015) and the one-stage RetinaNet (Lin et al., 2017b) frameworks for verification. We train models using the *trainval* split of MS COCO (Everingham et al., 2010) as training data and report the results in mean Average Precision (Lin et al., 2017a; Ren et al., 2015) on *minival* split. As shown in Table 5, the backbones obtained by our method (CafeNet 0.5×) can achieve higher performance compared to the uniform baseline (Uni 0.5×).

4.5 Ablation Studies

Effect of offset r for free channels. As described before, free channels freely locate in the local zone indicated by an offset r. Since we search on bins, r controls the size of search space; larger r implies more accurate evaluation of width but induces larger search space, corresponding to more computational cost. To investigate its effect, we compare the performance and time cost between different r on 0.5×-FLOPs MobileNetV2 and VGGNet on CIFAR-10 dataset. As shown in Fig.2(a), accuracy performance benefits from the increase of r while the larger r also aggravates the burden of training. For a trade-off between performance and time cost, we set $r = 1$ in all experiments. Note that in the training of CafeNet, larger r will have more sub-networks; however, identifying the one with the minimum loss only involves feed-forward calculation and does not increase training cost dramatically. More results refer to Appendix A.5.



(a) Performance of CafeNet under different offsets r

(b) Ablations of min-min optimization

Figure 2: (a) Performance with different offset r of free channels for the 0.5×-FLOPs MobileNetV2 and VGGNet on CIFAR-10 dataset. (b) Ablations of min-min optimization with 0.5×-FLOPs MobileNetV2 on CIFAR-10 dataset.

Effect of min-min optimization on supernet. During training CafeNet, we only optimize the sub-network with minimum training loss for each sampled width. To investigate the effect of this optimization strategy, suppose we have all τ iterations, then we implement min-min optimization only on the last $(1-\lambda)\cdot\tau$ iterations with $\lambda \in [0,1]$. For the first $\lambda\cdot\tau$ iterations, we simply optimize one sub-network randomly for each sampled width. As shown in Fig.2(b), our 0.5×-FLOPs MobileNetV2 on CIFAR-10 improves 0.92% accuracy from $\lambda = 1$ to $\lambda = 0$, which means that min-min optimization does help to better evaluate each width and boost the searching performance accordingly. We further record the performance of 1K sampled width; details refer to Appendix A.13.

Effect of performance improvement with more training and searching cost. Since CafeNet needs extra training and searching cost than the fixed weight sharing pattern, one natural question comes to *whether a fixed pattern can benefit from more training and searching cost and even surpass CafeNet as a result*. With this aim, we implement a fixed pattern (*i.e.*, $r = 0$ for CafeNet) with more times (1×∼3×) of training epochs and generations using evolutionary search to search for a 0.5×-FLOPs MobileNetV2 and VGGNet on CIFAR-10 dataset.

Table 6: Performance with more training and searching cost on CIFAR-10 dataset. Note CafeNet has 95.44% and 94.36% accuracy for 0.5×-FLOPs MobileNetV2 and VGGNet, respectively.

MobileNetV2				VGGNet			
Searching \ Training	1×	2×	3×	Searching \ Training	1×	2×	3×
1×	94.53%	94.57%	94.55%	1×	93.62%	93.65%	93.66%
2×	94.61%	94.67%	94.69%	2×	93.68%	93.73%	93.71%
3×	94.72%	94.75%	94.74%	3×	93.69%	93.66%	93.74%

As shown in Table 6, we can see that searching with more generations by evolutionary search can only improve the searching results slightly, while more times of training cost for supernet has almost no effect on the searching performance. By comparing with Table 3 and Fig.2(a), our CafeNet does efficiently improve the performance of searching results with the same or a little extra training and searching cost.

5 Conclusion

In this paper, we introduce a new weight sharing pattern for network width search. In detail, our locally free pattern enables each width is jointly indicated by its base channels and free channels, where free channels are used to better and more flexibly to evaluate the performance. Besides, we leverage FLOPs-sensitive bins to reduce the search space and allow FLOPs to distribute more evenly for different layers. CafeNet can be trained stochastically using min-min optimization and searches the optimal width by max-max selection. Extensive experiments have been conducted on benchmark ImageNet, CIFAR-10, CelebA, and COCO datasets to show the superiority of our CafeNet to other state-of-the-art methods.

REFERENCES

Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

Hesen Chen, Ming Lin, Xiuyu Sun, Qian Qi, Hao Li, and Rong Jin. Muffnet: Multi-layer feature federation for mobile deep learning. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

Zhengsu Chen, Jianwei Niu, Lingxi Xie, Xuefeng Liu, Longhui Wei, and Qi Tian. Network adjustment: Channel search guided by flops utilization ratio. *arXiv preprint arXiv:2004.02767*, 2020.

Ting-Wu Chin, Ruizhou Ding, Cha Zhang, and Diana Marculescu. Legr: Filter pruning via learned global ranking. *arXiv preprint arXiv:1904.12368*, 2019.

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*, pp. 248–255. Ieee, 2009.

Xiaohan Ding, Guiguang Ding, Yuchen Guo, Jungong Han, and Chenggang Yan. Approximated oracle filter pruning for destructive cnn width optimization. pp. 1607–1616, 2019.

Xuanyi Dong and Yi Yang. Network pruning via transformable architecture search. In *Advances in Neural Information Processing Systems*, pp. 759–770, 2019.

Xuanyi Dong, Junshi Huang, Yi Yang, and Shuicheng Yan. More is less: A more complicated network with less inference complexity. pp. 1895–1903, 2017.

Mark Everingham, Luc Van Gool, Christopher K I Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2): 303–338, 2010.

Xitong Gao, Yiren Zhao, Lukasz Dudziak, Robert Mullins, and Cheng-zhong Xu. Dynamic channel pruning: Feature boosting and suppression. *arXiv preprint arXiv:1810.05331*, 2018.

Shaopeng Guo, Yujie Wang, Quanquan Li, and Junjie Yan. Dmcp: Differentiable markov channel pruning for neural networks. *arXiv preprint arXiv:2005.03354*, 2020.

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.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

Peter JB Hancock. Pruning neural nets by genetic algorithm. In *Artificial Neural Networks*, pp. 991–994. Elsevier, 1992.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018a.

Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4340–4349, 2019a.

Yang He, Ping Liu, Linchao Zhu, and Yi Yang. Meta filter pruning to accelerate deep convolutional neural networks. *arXiv: Computer Vision and Pattern Recognition*, 2019b.

Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1389–1397, 2017.

Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 784–800, 2018b.

Yibo Hu, Xiang Wu, and Ran He. Tf-nas: Rethinking three search freedoms of latency-constrained differentiable neural architecture search. *arXiv preprint arXiv:2008.05314*, 2020.

Weizhe Hua, Yuan Zhou, Christopher M De Sa, Zhiru Zhang, and G Edward Suh. Channel gating neural networks. In *Advances in Neural Information Processing Systems*, pp. 1884–1894, 2019.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017.

Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 304–320, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55, 2014.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv: Computer Vision and Pattern Recognition*, 2016.

Tuanhui Li, Baoyuan Wu, Yujiu Yang, Yanbo Fan, Yong Zhang, and Wei Liu. Compressing convolutional neural networks via factorized convolutional filters. pp. 3977–3986, 2019.

Shaohui Lin, Rongrong Ji, Yuchao Li, Yongjian Wu, Feiyue Huang, and Baochang Zhang. Accelerating convolutional networks via global & dynamic filter pruning. In *IJCAI*, pp. 2425–2432, 2018.

Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. Towards optimal structured cnn pruning via generative adversarial learning. pp. 2790–2799, 2019.

Tsungyi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. pp. 936–944, 2017a.

Tsungyi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. pp. 2999–3007, 2017b.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*.

Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3296–3305, 2019a.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2736–2744, 2017.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019b.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. pp. 3730–3738, 2015.

Jian-Hao Luo and Jianxin Wu. Autopruner: An end-to-end trainable filter pruning method for efficient deep model inference. *arXiv preprint arXiv:1805.08941*, 2018.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. 2015:91–99, 2015.

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*, pp. 4510–4520, 2018.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. pp. 525–536, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. pp. 6105–6114, 2019.

Yulong Wang, Xiaolu Zhang, Lingxi Xie, Jun Zhou, Hang Su, Bo Zhang, and Xiaolin Hu. Pruning from scratch. *arXiv: Computer Vision and Pattern Recognition*, 2019.

Darrell Whitley, Timothy Starkweather, and Christopher Bogart. Genetic algorithms and neural networks: Optimizing connections and connectivity. *Parallel computing*, 14(3):347–361, 1990.

Mao Ye, Chengyue Gong, Lizhen Nie, Denny Zhou, Adam R Klivans, and Qiang Liu. Good subnetworks provably exist: Pruning via greedy forward selection. *arXiv: Learning*, 2020.

Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 2130–2141, 2019.

Jiahui Yu and Thomas Huang. Autoslim: Towards one-shot architecture search for channel numbers. *arXiv preprint arXiv:1903.11728*, 8, 2019.

Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 875–886, 2018.

A APPENDIX

We organize our appendix as follows. We explain the details of using CafeNet for training and searching in appendix A.1. In appendix A.2, we give details of our experiments w.r.t. different datasets. We report the implementation details of our CafeNet with Algorithm 1 in appendix A.3. Then, we introduce the corresponding relationship between names and references in appendix A.4. And we explore the efficiency of CafeNet in training one-shot supernet $\mathcal{N}$ A.5. We show the performance gain of each part of CafeNet in appendix A.6. We examine the effect of FLOPs-sensitive bins in appendix A.7. We explore more detailed experiments of ImageNet, CIFAR-10, and CelebA dataset in appendix A.8 and A.9. Then we show the visualization of searched network width in appendix A.11. In appendix A.12, we visualize the configuration of bins with EfficientNet-B0 and MobileNetV2. In addition, we investigate the effect of min-min optimization with MobileNetV2 and ResNet50 in appendix A.13. We study the impact of the multi-stage search with evolving bins on CIFAR-10 dataset in appendix A.14. We provide the estimation of the size of the search space in the multi-stage search in appendix A.15. Finally, the effect of the smallest bin size β in FLOPs-sensitive bins is shown in appendix A.16.

A.1 TRAINING AND SEARCHING WITH CAFENET

Training with min-min optimization. With the bin-based search space, we can train our CafeNet in a stochastic setting as Eq.(1), which simply samples a network width $\boldsymbol{c}$ first, and then optimizes the corresponding sub-network. Nevertheless, in CafeNet, a width is specified more freely by several sub-networks, and the performance of these sub-networks can be different to a great extent. According to this, we propose to indicate the performance of width by examining its sub-network with the best performance. Concretely, during the training of CafeNet, instead of randomly optimizing a sub-network, we optimize the sub-network with the smallest training loss. For a network width $\boldsymbol{c}$, we denote its corresponding sub-network set as $\mathbb{S}_{\boldsymbol{c}}$, thus for the CafeNet $\mathcal{N}$ with weights W, the training target of min-min optimization can be written as,

$$W^* = \arg\min_{W} \mathop{\mathbb{E}}_{\boldsymbol{c}\in U(\mathcal{C})} \left[\mathcal{L}_{train}(\boldsymbol{w}_{\boldsymbol{s}^*_{\boldsymbol{c}}}; \mathcal{N}, \boldsymbol{c}, \mathcal{D}_{tr})\right] \tag{11}$$

$$\text{s.t. } s^*_{\boldsymbol{c}} = \arg\min_{s\in\mathbb{S}_{\boldsymbol{c}}} \mathcal{L}_{train}(\boldsymbol{w}_s; \mathcal{N}, \boldsymbol{c}, \mathcal{D}_{tr}), \tag{12}$$

where $w_s \subset W$, denotes the weights of subnet s. To find the optimal sub-network $s^*_{\boldsymbol{c}}$, it needs to traverse all sub-networks and results in additional computational cost. However, this can be efficiently implemented by calculating the loss of all networks without backpropagation (*i.e.*, *no_grad* mode in PyTorch). And then, we only need to backward once using the target sub-network $s^*_{\boldsymbol{c}}$.

Searching with max-max selection. After the CafeNet $\mathcal{N}$ is trained, we can evaluate each network width by examining its performance (*e.g.* classification accuracy) on the validation dataset $\mathcal{D}_{val}$. Similar to the training of CafeNet, for a network width $\boldsymbol{c}$, we use the sub-network with the highest performance to indicate its performance. Then the searching amounts to a max-max selection problem

$$\boldsymbol{c}^* = \arg\max_{\boldsymbol{c}\in\mathcal{C}} \max_{s\in\mathbb{S}_{\boldsymbol{c}}} \text{Accuracy}(\boldsymbol{w}^*_s, W^*; \mathcal{N}^*, \mathcal{D}_{val}), \text{ s.t. FLOPs}(\boldsymbol{c}) \leq F_b. \tag{13}$$

Note that the inner $\max$ also needs to calculate the validation accuracy of all sub-networks. Nevertheless, since searching itself is much faster than the supernet training, the increased computational cost is subtle and acceptable in real practice.

A.2 DETAILS OF EXPERIMENTAL SETTINGS

In this section, we present the implementation details of our CafeNet w.r.t. experiments on various datasets. In general, for most networks except EfficientNet-B0, we use a SGD optimizer with momentum 0.9. While for EfficientNet-B0, we use RMSprop optimizer with epsilon set to 0.001. The parameters of β and α are initialized to 1 and 2, respectively, to determine the search space. For evolutionary search, we set the population and iteration size to 40 and 50, respectively. Besides, 2k network width are sampled for a random search. After searching, the optimized network width is trained from scratch for evaluation.

ImageNet and CIFAR-10 dataset. we set weight decay to 10^{-4} for ResNet50 (same setting is also adopted for ResNet34 and ResNet18) and VGGNet while 5×10^{-5} for MobileNetV2. The learning

rate is decayed with cosine strategy from 0.1 to 10^{-5} for MobileNetV2 and ResNet50 with 300 training epochs and a mini-batch size of 256. Besides, for ResNet50, we adopt the strategy of random erasing with a probability of 0.4. For VGGNet, we train for 400 epochs with batch size 128, and the learning rate is decayed by step strategy from 0.1 and divided by 10 at 150-th, 225-th, 300-th epoch. For EfficientNet-B0, the learning rate is initialized to 0.13 and decayed by 0.96 for every 3 epochs. We train EfficientNet-B0 for 300 epochs, of which the first 5 are warm-up epochs, and the weight decay is set to 1×10^{-5}.

CelebA dataset. We set the training epochs to 15 (10) for MobileNetV2 and ResNet18 with learning rate annealed from 0.1 to 10^{-5} by cosine strategy, and the batch size of these two networks are set to 64.

MS COCO dataset. We conduct object detection experiments using two popular frameworks Faster R-CNN with Feature Pyramid Networks (FPN) (Lin et al., 2017a) and RetinaNet (Lin et al., 2017b) on MS COCO dataset (Everingham et al., 2010). The *trainval35k* split is used for training and we report the mean Average Precision (mAP) on *minival* split. We train all the models using SGD for 12 epochs from ImageNet pretrained weights, and the initial learning rate is set to 0.08 with batch size 256, which decays 0.1 at 8-th and 11-th epoch.

A.3 ALGORITHM OF CAFENET

The details about CafeNet are presented in Algorithm 1. In specific, we search network width on CIFAR-10 dataset with 3 stages while others with 1 stage.

Algorithm 1 Locally free weight sharing for network width search

Input: The number of multi-stage $\mathcal{M}$. The training epochs of supernet $\mathcal{N}$. The smallest bin size β and the bin evloving speed α. Searching methods $\mathcal{A}$. Training dataset $\mathcal{D}_{tr}$. Validation dataset $\mathcal{D}_{val}$.

```
while stage < M do
    initialize the supernet N and bins groups.
    while epochs < E do
        randomly assign the training and backwards path
        updating the corresponding path of bin groups with dataset D_tr.
    end while
    update β = β / α
end while
search the optimized network width within supernet N by searching methods A (as illustrated in section 3.3).
train the optimized network width from scratch for evaluation.
```

Output: the optimized network width with evaluation results

A.4 INDEX AND LITERATURE

In order to simplify the correspondence between names and references in all tables, we have listed the names used and their corresponding papers as follows:

Table 7: The names used in tables and their corresponding papers.

Names	papers	Names	papers	Names	papers
ResNet	(He et al., 2016)	AutoSlim	(Yu & Huang, 2019)	Rethinking	(Liu et al., 2019b)
MetaPruning	(Liu et al., 2019a)	MIL	(Dong et al., 2017)	LEGR	(Chin et al., 2019)
PF	(Li et al., 2016)	GBN	(You et al., 2019)	CNN-FCF	(Li et al., 2019)
SFP	(He et al., 2018a)	LEGR	(Chin et al., 2019)	FPGM	(He et al., 2019a)
TAS	(Dong & Yang, 2019)	GS	(Ye et al., 2020)	AutoPruner	(Luo & Wu, 2018)
CGNet	(Hua et al., 2019)	MobileNetV2	(Sandler et al., 2018)	NA	(Chen et al., 2020)
MFP	(He et al., 2019b)	AMC	(He et al., 2018b)	DMCP	(Guo et al., 2020)
MuffNet	(Chen et al., 2019)	DCP	(Zhuang et al., 2018)	FBS	(Gao et al., 2018)
VGGNet	(Simonyan & Zisserman, 2014)	GAL	(Lin et al., 2019)	Sliming	(Liu et al., 2017)
PS	(Wang et al., 2019)	AOFP	(Ding et al., 2019)		

A.5 THE EFFICIENCY OF CAFENET IN TRAINING ONE-SHOT SUPERNET $\mathcal{N}$

To investigate the searching efficiency of CafeNet, we examine the time cost of training 1 epoch with different radius r. As shown in Table 8, the training time of CafeNet raised with an increase of r, which is because r represents the preset allowed offset for free channels, thus increasing the times of forwarding for each batch of information. However, since CafeNet involves multi-times forward and once backward and the time cost of forwarding is much shorter than the backward, the overall training efficiency will not be greatly affected. To balance the performance and search efficiency, we set $r = 1$ for all experiments with CafeNet.

Table 8: The efficiency of CafeNet in training one-shot supernet $\mathcal{N}$.

CIFAR-10			ImageNet		
Local free	MobileNetV2	VGGNet	Local free	MobileNetV2	ResNet50
r=0	51.3s	61.6s	r=0	428.4s	748.1s
r=1	61.1s	72.4s	r=1	520.6s	954.3s
r=2	95.2s	109.7s	r=2	832.5s	1569.6s
r=3	203.6s	258.4s	r=3	1872.3s	3584.9s

A.6 PERFORMANCE GAIN OF EACH PART IN CAFENET

To investigate the performance gain of each part in CafeNet, we conduct experiments to search for $0.5\times$ FLOPs MobileNetV2 and $0.5\times$ FLOPs VGGNet on CIFAR-10 dataset and report their Top-1 accuracy. Since min-min optimization and max-max selection are always used simultaneously, for clarity, we abbreviate them as min-min optimization. As shown in Table 9, with evolutionary search, CafeNet can enjoy a performance gain of 0.91% (0.74%) accuracy on MobileNetV2 (VGGNet). Besides, similar results can also be obtained through the greedy search, which exactly shows that our CafeNet can boost to search for a decent network width.

Table 9: Performance of searched $0.5\times$ FLOPs MobileNetV2 and VGGNet on CIFAR-10 dataset with different supernets and searching methods.

supernet			searching			models	
FLOPs-sensitive bins	Locally free weight sharing pattern	min-min optimization	multi-stage search	greedy search	evolutionary search	MobileNetV2	VGGNet
				✓		94.27%	93.49%
✓				✓		94.51%	93.65%
✓	✓	✓		✓		94.76%	93.89%
					✓	94.53%	93.62%
✓					✓	94.71%	93.74%
✓	✓	✓			✓	95.03%	93.95%
✓	✓	✓	✓		✓	95.44%	94.36%

A.7 EFFECT OF FLOPS-SENSITIVE BINS

Our proposed FLOPs-sensitive bins aim to reduce the search space by allowing the FLOPs to distribute more evenly over layers for the search unit. To explore the effect of FLOPs-sensitive bins as a search unit compared with the uniform unit (uniform channel group), we search for network width under these two search units on CIFAR-10 dataset with MobileNetV2 and VGGNet, as shown in Table 10. In detail, compared with the uniform unit, higher accuracy under various FLOPs budgets can be achieved by leveraging FLOPs-sensitive bins.

Table 10: Performance comparison of two forming search unit methods, *i.e.*, baseline uniform groups, and FLOPs-sensitive bins.

VGGNet			MobileNetV2		
FLOPs	Uniform groups	**FLOPs-sensitive bins**	FLOPs	Uniform groups	**FLOPs-sensitive bins**
189M	94.12%	**94.36%**	188M	95.37%	**95.56%**
154M	93.87%	**94.23%**	144M	95.12%	**95.44%**
115M	93.64%	**94.01%**	44M	94.99%	**95.31%**
76M	93.25%	**93.67%**	28M	93.62%	**94.11%**

A.8 MORE DETAILED RESULTS OF IMAGENET DATASET FOR TABLE 1

To explore the effectiveness of CafeNet, we also implement our algorithm on ResNet34 and ResNet18 with the same training strategy as ResNet50. The original ResNet34 and ResNet18 has 21.8M, 11.7M and 3.6G, 1.8G FLOPs with 74.9%, 71.5% Top-1 accuracy, respectively. The detailed results are reported in Table 11, our 0.75× FLOPs ResNet34 and ResNet18 can achieve close performance to the origin model. Besides, under the tiny FLOPs budget (*i.e.*, 10% FLOPs), CafeNet can outperform the unform baseline by a large margin.

Table 11: Performance comparison of ResNet50, ResNet34, ResNet18 and MobileNetV2 on ImageNet. References of baseline methods are summarized in appendix A.4.

ResNet50					
	Methods	FLOPs	Param	Top-1	Top-5
3G	AutoSlim	3.0G	23.1M	76.0%	-
	MetaPruning	3.0G	-	76.2%	-
	LEGR	3.0G	-	76.2%	-
	Uniform	3.0G	19.1M	75.9%	93.0%
	Random	3.0G	-	75.2%	92.5%
	CafeNet-R	3.0G	22.6M	**77.1%**	94.3%
	CafeNet-E	3.0G	23.8M	**77.4%**	94.5%
2G	GBN	2.4G	31.8M	76.2%	92.8%
	SFP	2.4G	-	74.6%	92.1%
	LEGR	2.4G	-	75.7%	92.7%
	FPGM	2.4G	-	75.6%	92.6%
	TAS	2.3G	-	76.2%	93.1%
	MetaPruning	2.0G	-	75.4%	-
	AutoSlim	2.0G	20.6M	75.6%	-
	Uniform	2.0G	13.3M	75.1%	92.7%
	Random	2.0G	-	74.6%	92.2%
	CafeNet-R	2.0G	19.1M	**76.5%**	93.1%
	CafeNet-E	2.0G	18.4M	**76.9%**	93.3%
1G	AutoPruner	1.4G	-	73.1%	91.3%
	MetaPruning	1.0G	-	73.4%	-
	AutoSlim	1.0G	-	74.0%	-
	Uniform	1.0G	6.6M	73.1%	91.8%
	Random	1.0G	-	72.2%	91.4%
	CafeNet-R	1.0G	11.2M	**74.9%**	92.3%
	CafeNet-E	1.0G	12M	**75.3%**	92.6%
570M	AutoSlim	570M	-	72.2%	-
	Uniform	570M	4.0M	71.6%	90.6%
	Random	570M	-	69.4%	90.3%
	CafeNet-R	570M	11.3M	**72.7%**	90.9%
	CafeNet-E	570M	12.0M	**73.3%**	91.2%

ResNet34					
	Methods	FLOPs	Param	Top-1	Top-5
2.7G	Rethinking	2.79G	-	72.9%	-
	MIL	2.75G	-	73.0%	-
	PF	2.79G	-	72.1%	-
	Uniform	2.7G	-	72.2%	90.9%
	Random	2.7G	-	71.6%	90.8%
	CafeNet-R	2.7G	18.6M	**74.4%**	92.0%
	CafeNet-E	2.7G	20.4M	**74.8%**	92.3%
	CNN-FCF	2.7G	15.9M	73.6%	91.5%
	CafeNet-R	2.5G	19.3M	**74.0%**	91.9%
	CafeNet-E	2.5G	20.2M	**74.5%**	92.1%
1.8G	FPGM	2.2G	-	72.5%	-
	SFP	2.2G	-	71.8%	90.3%
	CNN-FCF	2.2G	12.6M	72.8%	91.0%
	GS	2.1G	-	72.9%	-
	Uniform	1.8G	-	71.6%	90.3%
	Random	1.8G	-	71.1%	89.9%
	CafeNet-R	1.8G	17.2M	**73.1%**	91.4%
	CafeNet-E	1.8G	16.9M	**73.4%**	91.5%
	CGNet	1.8G	-	71.3%	-
	CNN-FCF	1.7G	9.6M	71.3%	90.2%
1G-	CNN-FCF	1.2G	7.1M	69.7%	89.3%
	CGNet	1.2G	-	70.2%	-
	Uniform	0.9G	-	69.5%	89.4%
	Random	0.9G	-	69.1%	88.9%
	CafeNet-R	0.9G	10.1M	**71.8%**	89.5%
	CafeNet-E	0.9G	9.8M	**72.1%**	89.8%
	Uniform	0.36G	-	59.9%	82.3%
	Random	0.36G	-	56.2%	80.6%
	CafeNet-R	0.36G	3.5M	**63.3%**	85.2%
	CafeNet-E	0.36G	3.6M	**64.0%**	85.4%

MobileNetV2					
	Methods	FLOPs	Param	Top-1	Top-5
200M	MetaPruning	217M	-	71.2%	-
	LEGR	210M	-	71.4%	-
	AutoSlim	207M	4.1M	73.0%	-
	Uniform	217M	2.7M	70.9%	89.4%
	Random	217M	-	70.3%	89.1%
	CafeNet-R	217M	3.0M	**73.3%**	91.1%
	CafeNet-E	217M	3.3M	**73.4%**	91.2%
150M	LEGR	150M	-	70.8%	-
	TAS	150M	-	70.9%	-
	AMC	150M	-	70.8%	-
	LEGR	150M	-	69.4%	-
	MuffNet	149M	-	63.7%	-
	Uniform	150M	2.0M	69.3%	88.9%
	Random	150M	-	68.8%	88.7%
	CafeNet-R	150M	2.7M	**71.9%**	90.0%
	CafeNet-E	150M	3.0M	**72.4%**	90.4%
100M	MetaPruning	105M	-	65.0%	-
	Uniform	105M	1.5M	65.1%	89.6%
	Random	105M	-	63.9%	89.2%
	CafeNet-R	106M	2.2M	**68.2%**	88.2%
	CafeNet-E	106M	2.1M	**68.7%**	88.5%
	MuffNet	50M	-	50.3%	-
	MetaPruning	43M	-	58.3%	-
	Uniform	50M	0.9M	59.7%	82.0%
	Random	50M	-	57.4%	81.2%
	CafeNet-R	50M	1.7M	**64.3%**	85.2%
	CafeNet-E	50M	1.6M	**64.9%**	85.4%

ResNet18					
	Methods	FLOPs	Param	Top-1	Top-5
1.2G	TAS	1.2G	-	69.2%	89.2%
	MIL	1.2G	-	66.3%	86.9%
	Uniform	1.2G	8.5M	68.8%	88.5%
	Random	1.2G	-	68.4%	88.1%
	CafeNet-R	1.2G	11.5M	**70.8%**	89.8%
	CafeNet-E	1.2G	11.3M	**71.2%**	89.9%
1G	NA	1.17G	-	69.4%	88.71%
	SFP	1.05G	-	67.1%	87.8%
	MFP	1.05G	-	68.3%	88.3%
	DMCP	1.04G	-	69.2%	-
	FPGM	1.04G	-	68.4%	88.5%
	DCP	0.96G	-	67.4%	87.6%
	CGNet	0.94G	-	68.8%	-
	MFP	0.9G	-	67.1%	87.5%
	FBS	0.9G	-	68.2%	88.2%
	Uniform	0.9G	6.0M	67.1%	87.5%
	Random	0.9G	-	66.7%	87.1%
	CafeNet-R	0.9G	9.7M	**69.6%**	88.8%
	CafeNet-E	0.9G	10.2M	**69.8%**	89.0%
0.45G-	Uniform	450M	2.9M	61.6%	83.6%
	Random	450M	-	59.8%	82.3%
	CafeNet-R	450M	5.0M	**65.2%**	86.0%
	CafeNet-E	450M	5.8M	**65.6%**	86.2%
	Uniform	180M	1.1M	53.7%	77.5%
	Random	180M	-	51.6%	76.9%
	CafeNet-R	180M	1.9M	**57.8%**	81.7%
	CafeNet-E	180M	2.0M	**58.4%**	81.9%

A.9 MORE DETAILED RESULTS OF CIFAR-10 AND CELEBA DATASET FOR TABLE 4 AND 5

To examine the hyperparameters setting with CIFAR-10 and CelebA dataset, we also explore the searching results from two vanilla baselines named Random and Uniform, as described in Section 4. The detailed results on CIFAR-10 dataset and CelebA dataset are reported in Table 12 and Table 13, respectively. Besides, for CelebA dataset, the layer widths of fully-connection(FC) layers are not implemented to be searched and keep the same as (Sener & Koltun, 2018), since they only takes a tiny amount of FLOPs to the network.

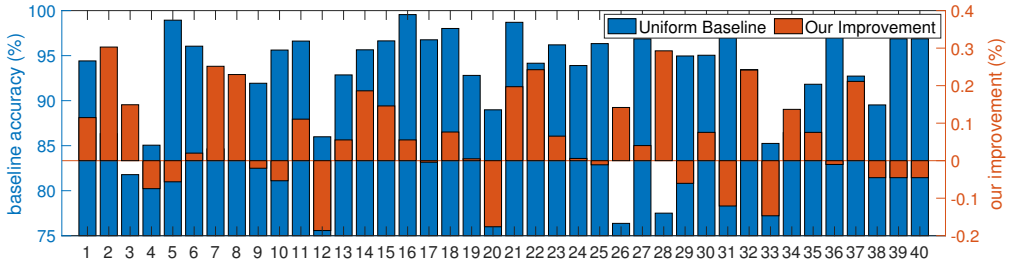
Table 12: Performance comparison of MobileNetV2 and VGGNet on CIFAR-10 dataset.

MobileNetV2					VGGNet				
Groups	Methods	FLOPs	Parameters	Accuracy	Groups	Methods	FLOPs	Parameters	Accuracy
200M	DCP	218M	1.7M	94.75%	200M	GAL	190M	-	93.80%
	Uniform	188M	1.4M	94.57%		DCP	199M	10.4M	94.16%
	Random	188M	-	94.20%		Sliming	199M	10.4M	93.80%
	CafeNet-R	188M	1.4M	**95.44%**		Uniform	189M	9.5M	93.37%
	CafeNet-E	188M	1.5M	**95.56%**		Random	189M	-	93.06%
144M	MuffNet	175M	-	94.71%		**CafeNet-R**	189M	8.3M	**94.27%**
	Uniform	144M	1.1M	94.28%		**CafeNet-E**	189M	8.0M	**94.36%**
	Random	144M	-	93.76%	100M+	PS	156M	-	93.63%
	CafeNet-R	144M	1.2M	**95.28%**		Uniform	154M	7.7M	93.11%
	CafeNet-E	144M	1.1M	**95.44%**		Random	154M	-	92.86%
44M	AutoSlim	88M	1.5M	93.20%		**CafeNet-R**	154M	3.4M	**94.09%**
	AutoSlim	59M	0.7M	93.00%		**CafeNet-E**	154M	3.1M	**94.23%**
	MuffNet	45M	-	93.12%		AOFP	124M	-	93.84%
	Uniform	44M	0.3M	92.72%		Uniform	115M	5.9M	92.95%
	Random	44M	-	92.24%		Random	115M	-	92.77%
	CafeNet-R	44M	0.4M	**95.16%**		**CafeNet-R**	115M	2.4M	**93.87%**
	CafeNet-E	44M	0.4M	**95.31%**		**CafeNet-E**	115M	2.1M	**94.01%**
28M	AutoSlim	28M	0.3M	92.00%	76M	CGNets	92M	-	92.88%
	Uniform	28M	0.3M	91.87%		Uniform	76M	3.9M	92.32%
	Random	28M	-	91.36%		Random	76M	-	91.67%
	CafeNet-R	28M	0.2M	**93.87%**		**CafeNet-R**	76M	2.2M	**93.36%**
	CafeNet-E	28M	0.2M	**94.11%**		**CafeNet-E**	76M	1.4M	**93.67%**

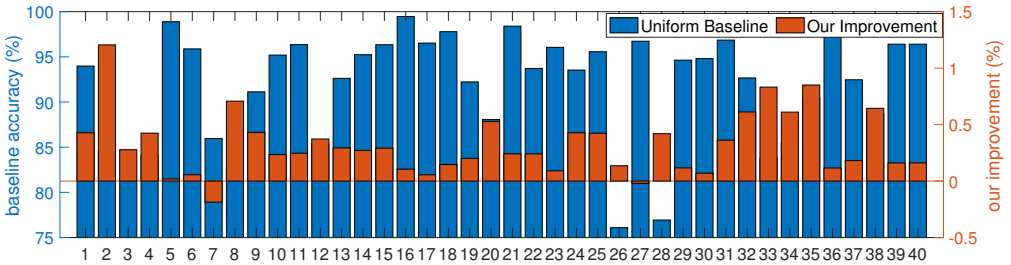
Table 13: Performance comparison of MobileNetV2 and ResNet18 on CelebA dataset.

MobileNetV2				ResNet18			
Methods	FLOPs	Parameters	Accuracy	Methods	FLOPs	Parameters	Accuracy
Uniform	162M	-	91.97%	Uniform	1G	-	92.03%
Random	162M	-	91.76%	Random	1G	-	91.84%
CafeNet-R	162M	1.7M	**92.12%**	**CafeNet-R**	1G	1.7M	**92.17%**
CafeNet-E	162M	1.8M	**92.19%**	**CafeNet-E**	1G	1.8M	**92.25%**
Uniform	106M	-	91.92%	Uniform	619M	-	91.93%
Random	106M	-	91.63%	Random	619M	-	91.67%
CafeNet-R	106M	1.2M	**92.09%**	**CafeNet-R**	619M	0.4M	**92.13%**
CafeNet-E	106M	1.2M	**92.16%**	**CafeNet-E**	619M	0.5M	**92.18%**
Uniform	51M	-	91.73%	Uniform	316M	-	91.79%
Random	51M	-	91.52%	Random	316M	-	91.62%
CafeNet-R	51M	0.6M	**92.03%**	**CafeNet-R**	316M	3.1M	**92.07%**
CafeNet-E	51M	0.5M	**92.13%**	**CafeNet-E**	316M	3.4M	**92.16%**
Uniform	21M	-	91.63%	Uniform	130M	-	91.66%
Random	21M	-	91.42%	Random	130M	-	91.51%
CafeNet-R	21M	0.2M	**91.71%**	**CafeNet-R**	130M	1.2M	**91.83%**
CafeNet-E	21M	0.2M	**91.85%**	**CafeNet-E**	130M	1.0M	**91.92%**

Classification results for all 40 labels on CelebA dataset. As shown in Fig.3, our searched network width has better performance in comparing to the uniform baseline, especially for small FLOPs budget and labels that are difficult to be classified.



(a) Performance of 50% FLOPs MobileNetV2 w.r.t.40 labels on CelebA dataset.



(b) Performance of 10% FLOPs MobileNetV2 w.r.t.40 labels on CelebA dataset.

Figure 3: Performance of 50% MobileNetV2 w.r.t.40 labels on CelebA dataset. The blue bar refers to the performance of network width searched by CafeNet, while the red bar indicates the performance gap in comparison to the uniform baseline.

A.10 MORE EXPERIMENTAL RESULTS WITH ALIGNED TRAINING RECIPES OF AUTOSLIM YU & HUANG (2019)

To examine the performance of our searched network width, we retrain the searched width of MobileNetV2 and ResNet50 with aligned training recipes of AutoSlim Yu & Huang (2019). In detail, we report the retraining results of width with similar FLOPs to AutoSlim in Table 14.

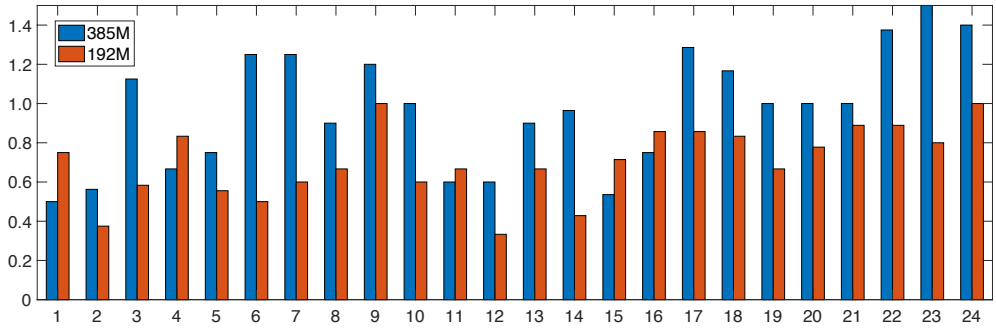
Table 14: Performance of MobileNetV2 and ResNet50 with aligned training recipes of AutoSlim Yu & Huang (2019).

ResNet50				
Methods	FLOPs	Parameters	Top-1	Top-5
AutoSlim	3.0G	23.1M	76.0%	-
CafeNet-R	3.0G	22.6M	76.2%	93.1%
CafeNet-E	3.0G	23.8M	76.4%	93.3%
AutoSlim	1G	-	74.0%	-
CafeNet-R	1G	11.2M	74.2%	91.9%
CafeNet-E	1G	12M	74.5%	92.1%
AutoSlim	570M	-	72.2%	-
CafeNet-R	570M	11.3M	72.1%	90.7%
CafeNet-E	570M	12M	72.6%	91.0%
MobileNetV2				
AutoSlim	207M	4.1M	73.0%	-
CafeNet-R	217M	3.0M	73.1%	90.9%
CafeNet-E	217M	3.3M	73.2%	91.1%

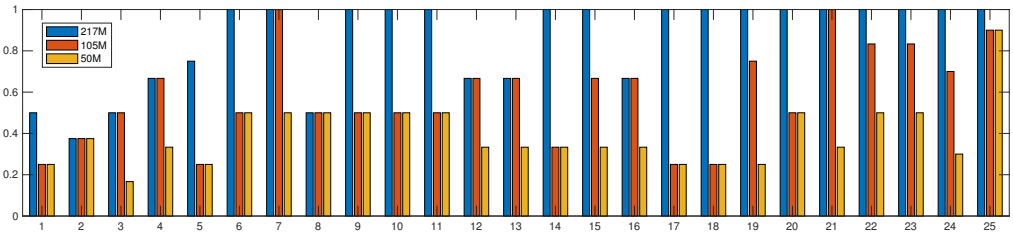
As Table 14 shows, with similar FLOPs and the same training recipes, CafeNet can search the width with higher accuracy, which proves the effectiveness of our method.

A.11 VISUALIZATION OF SEARCHED NETWORK WIDTH

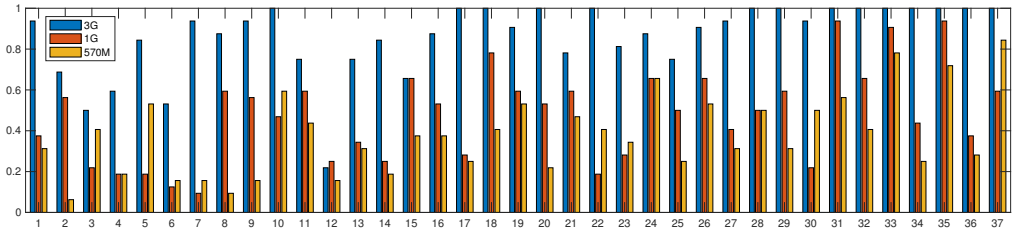
For intuitively understanding, we visualize our searched networks with ImageNet dataset, as shown in Fig.4. For clarity, we show the retained ratio of network width from the original model. Note that, EfficientNet-B0, ResNet50, and MobileNetV2 have SE block, skipping, or depthwise layers; we merged these layers, which are required to have the same network width for visual clarity. For EfficientNet-B0, we also did not show the channel of the SE block because these layers are fixed during the network width search.



(a) Network width of searched EfficientNet-B0 on ImageNet dataset.



(b) Network width of searched MobileNetV2 on ImageNet dataset.



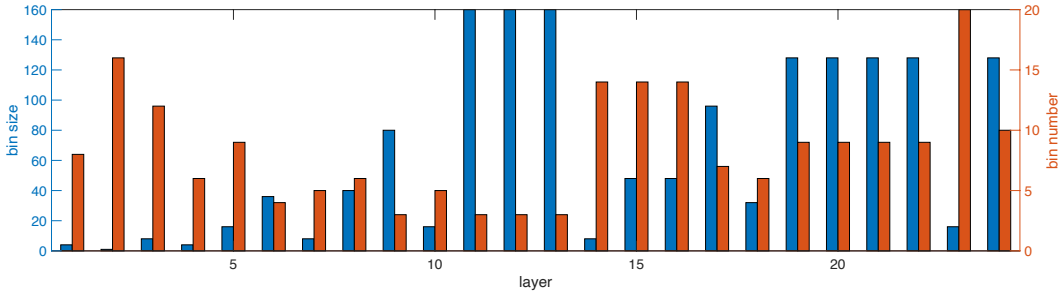
(c) Network width of searched ResNet50 on ImageNet dataset.

Figure 4: Visualization of Pruned models.

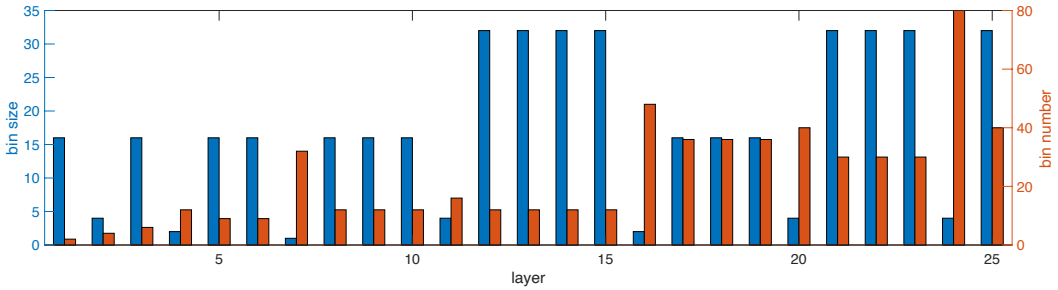
For ImageNet dataset, from Fig.4, we can see that layer width in skipping layers are more inclined to be preserved, which means skipping layers are very useful for classification. Besides, with a large FLOPs budget, width in the $1\times$ 1 convolution layers shrink more than in $3\times$ 3 convolution layers, which means $1\times$ 1 convolution may contribute less to classification performance. However, we observe an opposite phenomenon for width pruned from small FLOPs budgets, which implies the network is forced to use more $1\times$ 1 convolution layers to extract information from feature maps instead of $3\times$ 3 convolution layers.

A.12 VISUALIZATION OF BINS

For intuitively understanding of FLOPs-sensitive bins, as shown in Fig.5, we visualize the number of channels contained in each bin (bin size) and the number of bins in each layer (bin number) with MobileNetV2 and VGGNet. Similar to the visualization of the searched network width, we merged the SE blocks, skipping, and depthwise layers for clarity. To make the bins from all layers have a similar contribution to FLOPs, the number of bins varies significantly between layers. For EfficientNet-B0, the size of bin changes according to the different selected blocks (*i.e.*, blocks with 3×3 or 5×5 convolution kernel size), *i.e.*, the convolution of 3×3 or 5×5, and the size of bin becomes larger at the middle and the end of the network. For MobileNetV2, since only 3×3 convolution is involved, the bin size changes more regularly, and the number of bins gradually increases from the front to the end of the network.



(a) EfficientNet-B0



(b) MobileNetV2

Figure 5: Visualization of bin size and bin number of CafeNet.

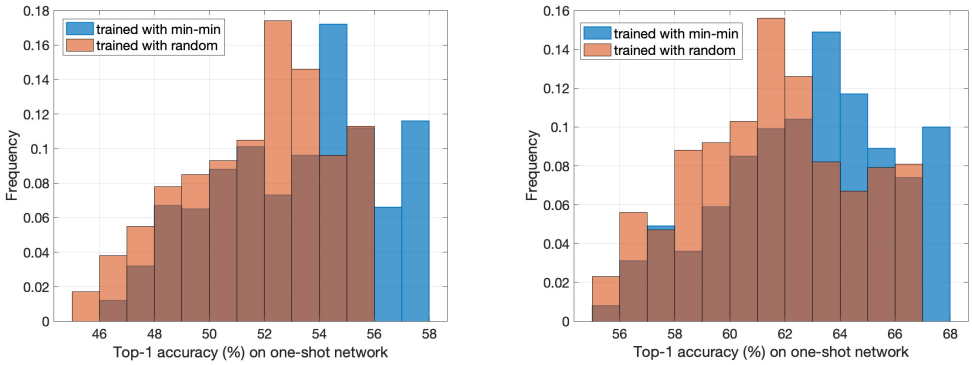
A.13 EFFECT OF MIN-MIN OPTIMIZATION IN EQ.(11)~(13)

To train the supernet, we investigate random optimization and min-min optimization with 50% FLOPs MobileNetV2 and ResNet50 on ImageNet dataset, for instance. In detail, based on the trained supernet, we use the evolution search to obtain the optimized network width and record 1000 network width for both random optimization and min-min optimization, and report the results by a statistical distribution histogram, as shown in Fig.6. During sampling, the average performance of network width with our min-min optimization achieved 62.4% (52.7%) Top-1 accuracy with ResNet50 (MobileNetV2), which is 1.4% (1.3%) better than results from random optimization. In addition, width with the best performance is trained from scratch for evolution, as shown in Table 15.

Table 15: Performance comparison of weights updating methods.

0.5×ResNet50			0.5×MobileNetV2		
Method	Sampling	CafeNet	Method	Sampling	CafeNet
Random	67.14%	76.34%	Random	56.22%	71.97%
min-min	**68.56%**	**76.92%**	**min-min**	**58.37%**	**72.43%**

In Table 15, we record the accuracy of Top-1 performance network width in supernet as Sampling, and than we training from scratch the width to report its Top-1 accuracy as CafeNet.



(a) Histogram of Top-1 accuracy by evolutionary with MobileNetV2.

(b) Histogram of Top-1 accuracy by evolutionary with ResNet50.

Figure 6: Histogram of Top-1 accuracy of network width trained from min-min optimization and random optimization by evolutionary w.r.t. MobileNetV2 and ResNet50.

A.14 EFFECT OF MULTI-STAGE SEARCH WITH EVOLVING BINS

To examine the effect of multi-stage search with evolving bins, we report the Top-1 accuracy of MobileNetV2 on CIFAR-10 dataset with different training stages and the bin evolving speed α in Section 3.4, the results are summarized in Table 16. In detail, for CIFAR-10 dataset, our algorithm achieves superior performance with α set to 2 and searching in 3 stages, which indicates bin evolving strategy can lead to better results comparing with searching in one stage. However, when α is increased to 4, the performance of the searching results decreases a bit, which may be due to the larger search space that limits the ability of evolutionary search.

Table 16: Performance of 50% FLOPs MobileNetV2 and VGGNet on CIFAR-10 dataset with different training stages and bin evolving speed α.

Number of training stages	Bin evolving speed α	VGGNet	MobileNetV2
1	-	93.95%	95.03%
2	1	94.22%	95.31%
2	2	94.29%	95.41%
2	4	94.18%	95.29%
3	1	94.29%	95.36%
3	2	94.36%	95.44%
3	4	94.21%	95.31%

A.15 ESTIMATION OF THE SIZE OF SEARCH SPACE IN MULTI-STAGE SEARCH

Note that by leveraging the multi-stage searching strategy, we can further reduce the search space by following the searching "from coarse-grained to fine-grained" with evolving bins "from large to small". In this way, we directly investigate how the search space is reduced by the multi-stage searching strategy. We compare the search space of supernet under different training stages, bin size β, and bin evolving speed α in Table 17. To fairly compare the results with different training stages, we set the size of the bin unit to be the same in Table 17. And the search space is defined as the product of the number of bins in all layers.

$$\mathcal{B}^{s}_{space} = \sum_{j=1}^{s} \prod_{i=1}^{L} \frac{Channel_i^j}{b_i^j} \tag{14}$$

Where $Channel_i^j$ represents the total channels at layer i in stage j, and b_i^j indicates the bin size of layer i in stage j. We examine the search space of 0.5× FLOPs MobileNetV2 on ImageNet dataset, as shown in Table 17.

Table 17: Search space of 0.5× FLOPs MobileNetV2 on ImageNet dataset.

0.5× MobileNetV2				
Stages	Bin size β	Evolve speed α	Search space	Percent
1	-	-	1.5×10^{27}	1
2	1	2	2.0×10^{19}	1.3×10^{-8}
2	2	4	4.8×10^{10}	3.0×10^{-17}
3	2	2	6.2×10^{10}	3.9×10^{-17}
4	4	2	7.3×10^{3}	4.6×10^{-24}

In detail, our search space reduced by more than $1/(1.3 \times 10^{-8})$ times from the original size (with one stage) by searching more than or equal to 2 stages. As a result, our proposed multi-stages search method can effectively reduce the search space.

A.16 Effect of smallest bin size β in FLOPs-sensitive bins

As formulated in Eq.(9), the total number of bins within each layer relies on the smallest bin size β. To examine the effect of β w.r.t. bin numbers and its corresponding performance, we perform experiments with MobileNetV2 on ImageNet dataset under 50% FLOPs budget, as shown in Table 18. In detail, the performance of searched network width first increases when β goes large but a little decreased after β surpasses 0.5, which may result from that a very small β induces a too huge search space thus is harmful to searching algorithms to find the optimized network width.

Table 18: Performance of ResNet50 and MobileNetV2 on ImageNet dataset w.r.t. different smallest bin size β for FLOPs-sensitive bins.

MobileNetV2			ResNet50		
Smallest bin size β	Top-1	Top-5	Smallest bin size β	Top-1	Top-5
4	72.22%	90.22%	4	76.43%	93.01%
2	72.32%	90.25%	2	76.73%	93.17%
1	72.40%	90.38%	1	76.91%	93.33%
0.5	72.34%	90.29%	0.5	76.82%	93.24%
0.25	72.28%	90.31%	0.25	76.67%	93.14%