Network Amplification with Efficient MACs Allocation

Chuanjian Liu, Kai Han, An Xiao, Ying Nie, Wei Zhang, Yunhe Wang
Huawei Noah's Ark Lab
{liuchuanjian,kai.han,an.xiao,ying.nie,wz.zhang,yunhe.wang}@huawei.com

Abstract

Recent studies on deep convolutional neural networks present a simple paradigm of architecture design, i.e.*, models with more MACs typically achieve better accuracies, such as EfficientNet and RegNet. These works try to enlarge the network architecture with one unified rule by sampling and statistical methods. However, the rule is not prospective to the design of large networks because it is obtained from the experience of researchers on small network architectures. In this paper, we propose to enlarge the capacity of CNN models by fine-grained MACs allocation for the width, depth and resolution on the stage level. In particular, starting from a base small model, we gradually add extra channels, layers or resolution by using a dynamic programming manner. With step-by-step modifying the computations on different stages, the enlarged network will be equipped with optimal allocation and utilization of MACs. On EfficientNet, our method consistently outperforms the performance of the original scaling method. In particular, the proposed method is used to enlarge models sourced by GhostNet, we achieve state-of-the-art* 80.9% *and* 84.3% *ImageNet top-1 accuracies under the setting of 600M and 4.4B MACs, respectively.*

1. Introduction

Convolutional neural networks (CNNs) deliver state-of-the-art accuracy in many computer vision tasks such as image classification [12, 20], object detection [29], image super-resolution [18]. Most of deep CNNs are well designed with a predefined number of parameters and computational complexities. For example, ResNet [12] mainly consists of 5 configurations with 18, 34, 50, 101 and 152 layers. These CNNs have provided strong baselines for visual applications.

To improve the accuracy further, the general method is to scale up the CNN model. Three factors including depth, width and input resolution heavily affect the model size. A number of works propose to scale models by the depth [12, 32], width [41] or input image resolution [16].

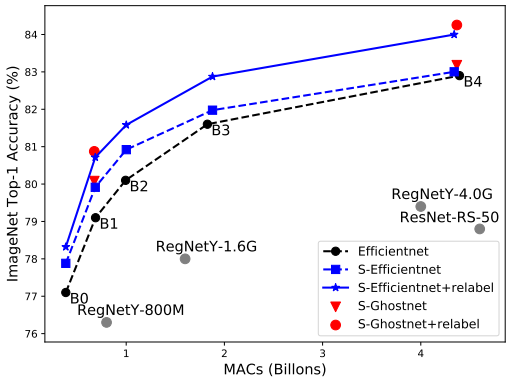

Figure 1. ImageNet classification results of our method. The black dash line is the original EfficientNet series. The blue dash line is searched S-EfficientNet and the blue solid line is S-EfficientNet with relabel trick. The red circle and triangle is the performance of GhostNet-based architectures.

These works usually consider only one dimension from depth, width and resolution, which leads to the imbalance in utilization of the computations or multiply-accumulate operations (MACs) . Simultaneously amplifying the width, depth and resolution can provide more flexible design space to find the high-performance models. Recently, several works focus on how to efficiently scale the three factors. EfficientNet [34] constructed one compound scaling formula to constrain the network width, depth and dimension. RegNet [26] studied the relationship between width and depth by exploring the network design spaces. These methods utilize a unified principle to scale the whole model, but ignore the differences between stages. Besides, the principles of these model scaling methods are acquired from a large number of small models, which is nonperceptible to the architecture of large models.

In this paper, we rethink the procedure of enlarging CNN models from the viewpoint of stage-wise computation resource allocation. Modern CNNs usually consists of several stages, where one *stage* contains all layers with the same spatial size of feature maps. In Figure 2, we present the computations of different stages for ResNet [12] and Ef-

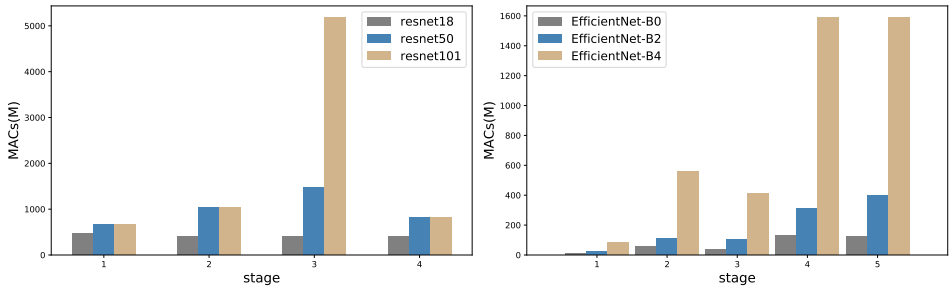

Figure 2. MACs of different stages of CNN models. Left figure presents the MACs of ResNet series. Right figure presents the MACs of EfficientNet series.

ficientNet [34]. Figure 2 left demonstrate the discrepancy between ResNet series. ResNet18 has balanced MACs for each stage, while ResNet50 and ResNet101 get more MACs in the intermediate stages but few MACs in the head and tail stages. Figure 2 right presents the allocation of MACs for EfficientNet-B0, EfficientNet-B2 and EfficientNet-B4. EfficientNet utilizes one unified model scaling principle for network width, depth and resolution, so different configurations of EfficientNet have the similar tendency of MACs on different stages. The later stages have far more MACs ($>$10 times) than the former and intermediate stages. This contrast declares that an universal rule of the computation allocation for different models is impractical. Neither the manual designed nor the unified rule is the solution of optimal computations allocation.

In this paper, we propose a network enlarging method based on the allocation of computations for each stage. In contrast to conventional unified principle, the method performs fine-grained search on the reallocation of computations. Given a baseline network, our goal is to enlarge it to the target MACs with the best configuration of depth, width and resolution for all stages. Under the assumption that the top-performing smaller CNNs are a proper subcomponent of the top-performing larger CNNs, we are able to enlarge CNNs step-by-step by using dynamic greedy network enlarging algorithm. For each iteration in our algorithm, 1) a series of candidate networks are constructed by searching width, depth and resolution of each stage under constrained MACs; 2) with fast performance evaluation method, the architecture with the best performance in this iteration is appended to the baseline model pool for next iteration. By gradually adding MACs at each iteration, we find the optimal architecture until achieving the target MACs. Experiment results on ImageNet classification task demonstrate the superiority of our proposed method. The searched network configurations can largely boost the performances of existing base models. For example, searched EfficientNet models by proposed method outperform the original EfficientNets by a large margin.

2. Related Work

Manual Network Design. In the early days after AlexNet [20], a large number of manually designed network architecture emerged. VGG [32] is the typical CNN architecture without any special connections, and deeper VGGnets get high accuracies. However, the degradation problem emerged for this type of very deep network. ResNet [12] with shortcut was proposed with higher accuracy and more layers. Except deeper network, wider network is another direction. WideResnet [41] has higher accuracy by adding channels for each layer in Resnet. Besides, a large number of light-weight networks are proposed in order to meet the demands of mobile devices. GoogLeNet [33], MobileNets [14,30], ShuffleNets [25,42] and GhostNet [11] are these type of networks. By setting one width scaling factor, the accuracy and MACs of Mobilenets and GhostNet are improved. The design pattern behind these networks was largely man-powered and focused on discovering new design choices that improve accuracy, *e.g.*, the use of deeper or wider models or shortcuts.

Automatic Network Design. Currently expert designed architectures are time-consuming. Because of this, there is a growth interest in automated neural architecture search (NAS) methods [8]. By now, NAS methods have outperformed manually designed architectures on some tasks such as image classification [13, 22, 23, 27, 44], object detection [10, 17, 36, 39] or semantic segmentation [21, 31]. Generally, more MACs means higher accuracy. Traditionally, researchers have already learned to change the depth, width or resolution of [24, 37] models. FBNetV2 [37] supports searches over spatial and channel dimensions except original microscopic architectures. JointPruning [24] proposed an effective method of pruning a network on three crucial aspects: spatial, depth and channel simultaneously. EfficientNet [34] showed that it was critical to balance all dimensions of network width/depth/resolution and proposed a simple yet effective compound scaling method in accordance with the results by random sampling. RegNet [26]

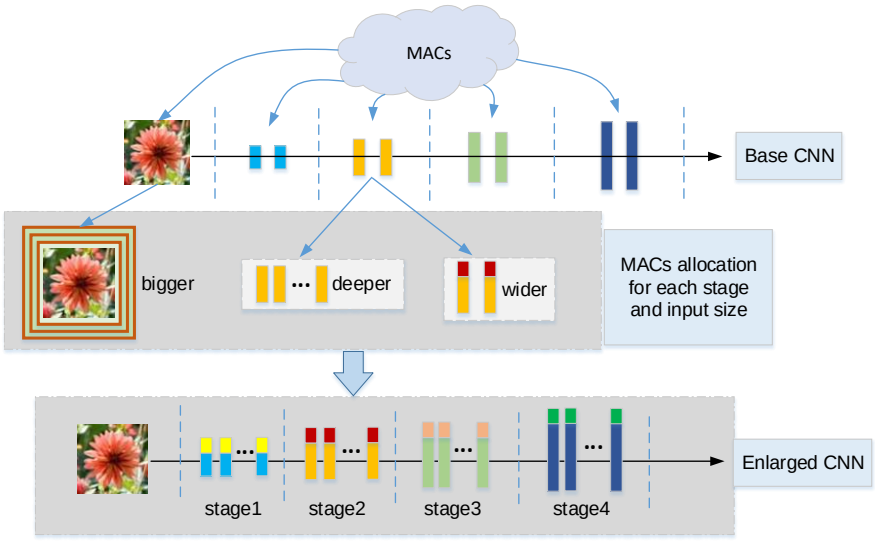

Figure 3. The framework for adjusting input resolution, width and depth for each stage. The surrounding box out of the input image means candidate input resolutions.

got several patterns by a huge number of experiments: 1. good network have increasing widths with stages; 2. the stage depths are likewise tend to increase for the best models, although not necessarily in the last stage. These methods construct rules from small networks, and use the rules to get various sizes of models, even very large models. In this paper, we optimize the allocation of MACs and get the specific model architecture under constrained MACs. During the expansion, the width, depth and input resolution are considered for each stage. Our intention is to maximize the utilization of MACs for the network.

3. Approach

In this section, we describe the proposed network enlarging method, which is based on the greedy allocation of MACs. Firstly, we define the goal of our method, which is to find the optimal depth and width of each stage and the input resolution. Secondly, we introduce the main algorithm of dynamic greedy network enlarging. Thirdly, we introduce how to evaluate the performance of candidate models efficiently.

3.1. Problem Definition

The modern CNN backbone architectures usually consists of a stem layer, network body and a head [12, 26, 34]. The main MACs and parameters lie in the part of network body, as typically the stem layer is a convolutional layer and the head is a fully-connected layer. Thus, in this paper we focus on the scaling strategy of network body. General network body consists of several *stages* [26], which are defined as a sequence of layers or blocks with the same spatial size or the same number of channels. For example, ResNet50 [12] body is composed of 4 stages. The *width* is defined as the number of channels and the *depth* is defined as the number of blocks or layers.

Scaling up convolution neural networks is widely used to achieve better accuracy. Network depth, width and input resolution are three key factors for model scaling. Deeper convolution neural networks capture richer and more complex features, and usually have high performance in contrast to shallow network. With the help of shortcuts, very deep network can be trained to convergence. However, the improvements on accuracy become smaller with the increase of depth. Another direction is scaling the width of network. More kernels mean more fine grained features can be learned. However, the MACs is squared with the network width. As a result, the network depth will be constrained and high level features maybe loss if a wider network is designed. Besides, EfficientNet [34] has showed that the accuracy quickly saturates when networks become wider. Higher resolution provides rich fine-grained information. In order to match high resolution, more powerful network is wanted. Deeper and wider network can acquire large receptive field and capture fine grained features together.

In conclusion, the depth, width and resolution of one network architecture are not independent. These three dimensions have various combinations. And one unified principle can not acquire the best configuration for all cases. In this paper, we decompose the network depth, width and resolution into stage depth, stage width and input resolution. This will maximize the utilization of computations for each stage and the whole network.

Given a base network $\mathbb{N}$ with L stages, width and depth are $\mathbf{w}, \mathbf{d} = (w_1, w_2, ..., w_L, d_1, d_2, ..., d_L)$, and input reso-

lution is $\mathbf{r}$. The objective is to acquire the network architecture with best performance by optimizing the allocation of target MACs T. Its mathematical form is:

$$\mathbf{r}^\star, \mathbf{d}^\star, \mathbf{w}^\star = \arg\max_{\mathbf{r},\mathbf{d},\mathbf{w}} \mathrm{ACC}_{\mathrm{val}}(\mathbb{N}(\mathbf{r}, \mathbf{d}, \mathbf{w}, \theta^\star)) \quad (1)$$

$$\text{s.t.} \quad |MACs(\mathbb{N}(\mathbf{r}, \mathbf{d}, \mathbf{w})) - T| \leq \delta \cdot T \quad (2)$$

$$\theta^\star = \arg\min_{\theta} \mathrm{LOSS}_{\mathrm{train}}(\mathbb{N}(\mathbf{r}, \mathbf{d}, \mathbf{w}, \theta)) \quad (3)$$

where θ is the trainable parameters of the network. $\mathrm{ACC}_{\mathrm{val}}$ denote the validation accuracy, T is the target threshold of MACs and δ is used to control the difference between the MACs of the searched model and the target MACs.

Search space. The search space contains input resolution, width and depth of all stages. Suppose a base network with L stages and configurations of width and depth $\mathbf{w}, \mathbf{d} = (w_1, w_2, ..., w_L, d_1, d_2, ..., d_L)$, and input resolution $\mathbf{r}$. The minimum growth size for width, depth and resolution is s_w, s_d and s_r, respectively. Under constrained target MACs, we have to search all combinations of width, depth for each stage and the network input resolution. The search space is huge. For example, ResNet18 contains 4 stages, and the minimum step-wise growth size is 1 for depth and 4 for width. If we constrain the search upper bound is 3 times in both depth and width for each stage. Then for first stage with 2 blocks and 64 channels, the candidate widths are $\{64, 68, ..., 192\}$, totally 65; and the candidate depths are $\{2, 3, ..., 6\}$, totally 5. The total number of combinations is $33 \times 65 \times 129 \times 257 \times 5 \times 5 \times 5 \times 5 \approx 4.4 \times 10^{10}$ without considering the variation of input size. Through reducing the target MACs of each iteration, the search space is dramatically reduced.

3.2. Algorithm of Efficient MACs Allocation

Figure 3 presents our framework. Our intention is to find the optimal architecture under constrained computations through probably allocation of computations to network depth, width and input resolution. So as to maximize the utilization of MACs, as shown in Eq. 1. For each stage i in the network, we try to find its optimal stage depth $\mathbf{d_i^\star}$, width $\mathbf{w_i^\star}$ and input size $\mathbf{r}^\star$. In the problem, $\mathbf{r}, \mathbf{d}, \mathbf{w}$ have discrete values and massive combinations. Directly search for the optimal combination $\mathbf{r}^\star, \mathbf{d}^\star, \mathbf{w}^\star$ is both time and resource consuming. Due to the extreme complexity, traditional mathematical optimization method is impracticable. So we turn to efficient neural architecture search method.

Finding the global optimal model is difficult in the massive search space, so we can smooth the procedure of search. Specifically, we decompose the optimal network architecture search problem into a series of optimal sub-network architecture search problem. We add the target MACs gradually in the iterative process. Then,for each iteration, we search the optimal width, depth and resolution under current target MACs.

Algorithm 1: Efficient MACs Allocation.

Result: Configurations $\mathbf{c}^\star = \mathbf{r}^\star, \mathbf{d}^\star, \mathbf{w}^\star$ with target MACs $\mathbf{T}$.

Initialization: Base network $\mathbb{N}$ with B_0 MACs and $\mathbf{L}$ stages: width $\mathbf{w^0} = (w_1^0, w_2^0, ..., w_L^0)$, depth $\mathbf{d^0} = (d_1^0, d_2^0, ..., d_L^0)$ and input resolution $\mathbf{r^0}$. Total dimension of search space is $(2L + 1)$. The target MACs of output network is $\mathbf{T}$ and the rate of error is δ. The search number is $\mathbf{N}$. Initialize the set of optimal sub-configurations as $\boldsymbol{S} = \{(\mathbf{r^0}, \mathbf{d^0}, \mathbf{w^0})\}$, the growth rate of resolution s_r;

```
while i ≤ N do
    current target MACs: T_i = B_0 * (T/B_0)^(i/N);
    current candidates C = [ ];
    for (r, d, w) in S do
        while MACs(N(r, d, w)) < T_i do
            r = r + s_r;
        end
        if |MACs(N(r, d, w)) − T_i| ≤ δ · T then
            C append (r, d, w)
        end
    end
    for (r, d, w) in S do
        for j in range(L) do
            while MACs(N(r, d, w)) < T_i do
                C' = proportional collection(d, w) as
                in Algorithm 2;
            end
            C extend C'
        end
    end
    ACC = [ ];
    for c in C do
        acc = performance evaluation(c) as
        stated in Sec. 3.3;
        ACC append acc;
    end
    index = arg max ACC;
    S append C_index;
    if i == N then
        c* = C_index;
    end
end
```

If sub-problems can be nested recursively inside larger problems, so that **dynamic programming** method is applicable. For this problem, the **current state** is defined as one set of optimal sub-network configurations(width, depth and resolution) searched before this iteration. In the process of search, the selection of decisions at each iteration depends

Algorithm 2: Proportional collection of width and depth.

Result: Candidates $\boldsymbol{C}$ with collected width w and depth d

Initialization: Target MACs T_i, configuration $(\mathbf{r}, \mathbf{d}, \mathbf{w})$, ratios set $\boldsymbol{P}$. The growth rate of depth and width is s_d and s_w, respectively, current stage j;

$d_j \in \mathbf{d}$; $w_j \in \mathbf{w}$;

for $p \in \boldsymbol{P}$ **do**

  $T_d = p * T_i$;

  while $MACs(\mathbb{N}(\mathbf{r}, \mathbf{d}, \mathbf{w})) \leq T_d$ **do**

    $d_j = d_j + s_d$

  end

  while $MACs(\mathbb{N}(\mathbf{r}, \mathbf{d}, \mathbf{w})) \leq T_i$ **do**

    $w_j = w_j + s_w$

  end

  if $|MACs(\mathbb{N}(\mathbf{r}, \mathbf{d}, \mathbf{w})) - T_i| \leq \delta \cdot T$ **then**

    $\boldsymbol{C}$ append $(\mathbf{r}, \mathbf{d}, \mathbf{w})$

  end

end

only on the current state. When the decision of each iteration is determined, a decision sequence is formed, which determines the final solution.

With the above description, we predefine the search iterations as N. For iteration i, we compute the current target MACs as T_i and search the optimal network configuration $\boldsymbol{A_i}$. Then we add $\boldsymbol{A_i}$ to the current best configurations to get $\boldsymbol{S_i} = \{\boldsymbol{A_0}, \boldsymbol{A_1}, ..., \boldsymbol{A_i}\}$. Then for the next iteration $(i+1)$, we compute the target MACs T_{i+1}. For each configuration in $\boldsymbol{S_i}$, we construct the candidate set $\boldsymbol{C_{i+1}}$ and evaluate each item in $\boldsymbol{C_{i+1}}$ to get the optimal configuration $\boldsymbol{A_{i+1}}$ of iteration $(i+1)$. The process is repeated until the predefined search number N achieved. The details of the algorithm are in Algorithm 1.

In order to reduce the number of candidate architectures in the process of acquiring candidates, we decompose the combinations of $(\mathbf{r}, \mathbf{w}, \mathbf{d})$ into $\mathbf{r}$ and the combinations of $(\mathbf{w}, \mathbf{d})$. In this process of searching the resolution $\mathbf{r}$, we fix the size of $(\mathbf{w}, \mathbf{d})$ and explore all feasible resolutions for all candidate in candidate set. Then we fix the size of resolution $\mathbf{r}$ and search all combinations of width and depth $(\mathbf{w}, \mathbf{d})$ for all candidate in candidate set. We take use of proportional control factor to assign the MACs between depth and width for each stage. Specifically, the ratios of MACs between depth and width are in one set $\boldsymbol{P}$. Under this setting, we search depth first and then width for each stage. The algorithm is illustrated in Algorithm 2.

In the algorithm, we use exponential increment of MACs in the process of search. This way make the changes of network more smoothly in contrast to uniform increment. For each iteration in Algorithm 1, in order to find the local optimal architecture configuration, we have to search and evaluate the candidate architectures. This step contains two targets: the first is to find the candidate architectures under limited increase of MACs; the second is to find the local optimal architectures with maximum validation accuracy.

3.3. Efficient Performance Estimation

To guide the search process, we have to estimate the performance of a given architecture. The most accurate method is to train the candidates on the whole training data and evaluate their performance on validation data. However, this way requires great computational demands in the order of thousands of GPU days. Developing methods for speeding up the process of performance estimation is crucial.

Inspired by the idea of network transformation [3, 4, 7], the knowledge of pretrained model could be transfer to the derived new architecture. With the help of this trick, there is no need to retrain all new architectures. We finetune the derived new architectures for several epochs, which accelerate the search process greatly. Next we further illustrate the efficient performance estimation process.

Directly training on the whole dataset is unacceptable. We turn to one proxy task to estimate performance. Including shorter training times [26], training on a subset of the data [19], on proxy data [44] or using less filters per layer and less cells [28]. These low-fidelity approximations reduce the cost, they also introduce bias in the performance estimation. Proxy data and simplified architecture have large deviation which leads to poor rank preservation.

In this section, we determine the optimal proxy task for performance estimation with empirical experiments. Firstly, we get the proxy sub-dataset by evaluating the performance of different sub-datasets. Secondly, the hyper-parameters of training are acquired with parameter search. Spearman's rank correlation coefficient ρ is a nonparametric measure of rank correlation, which is used as the measure of proxy task.

For the proxy sub-dataset, we create two sub-datasets ImageNet1000-100 and ImageNet100-500 by random selecting images from ImageNet. To evaluate these datasets, 12 network architectures with different width, depth and input sizes are generated on the basis of EfficientNet-B0. We train all the 12 networks and EfficientNet-B0 on the whole train set of ImageNet for 150 epochs, the Top-1 accuracies on the validation dataset are used as the comparison object. We finetune the 12 networks for different epochs. Besides, we train the 12 networks from scratch for few epochs. On ImageNet100-500, the average Spearman value is $\rho = 0.16$. On ImageNet1000-100, the average Spearman value is $\rho = 0.23$. So we choose ImageNet1000-100 as the proxy sub-dataset. More details are presented in the supplementary materials.

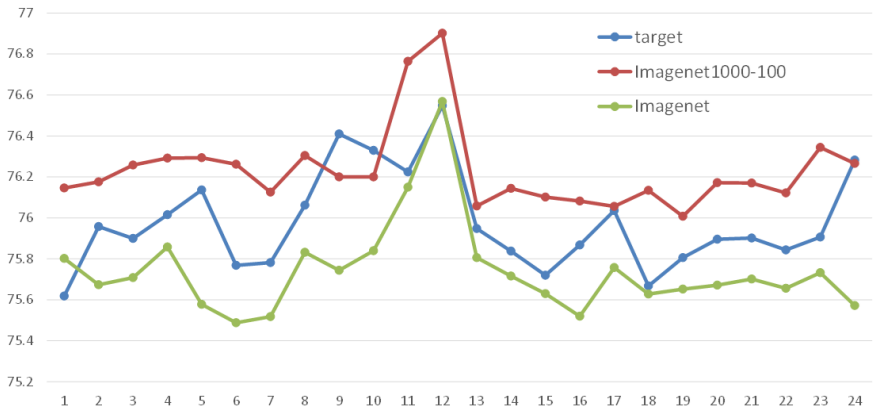

Figure 4. Correlation between the proxy and original task. The X-axis is the 24 networks with different configurations. The Y-axis is the top-1 accuracy. The blue line is the target. The red line trained on ImageNet1000-100 has Spearman value 0.57. The cyan line trained on ImageNet has spearman value 0.54. For visualization, we adjust the accuracy up by 27 and 10 for red line and cyan line, respectively.

After determining the proxy dataset, we try to improve the correlation between the proxy task and original task by searching the hyper-parameters. 24 network architectures with different width, depth and input sizes are generated on the basis of EfficientNet-B0. We train all of the 24 networks on the whole train set of ImageNet for 150 epochs, the Top-1 accuracies on the validation dataset are used as the comparison object. Two pretrained EfficientNet-B0 models on the ImageNet and ImageNet1000-100 are provided, respectively. The learning rate, mode of learning rate decay and training epochs are considered. Among these hyper-parameter combinations, the top-2 Spearman value ρ is 0.57 and 0.54, these values indicate moderate positive correlation. They both use cosine decay method and the initial learning rate is 0.01 for training 20 epochs. The difference is that the first use the ImageNet1000-100 pretrained model and the second use the ImageNet pretrained model. More details are presented in the supplementary materials. Figure. 4 presents the consistency of different networks. In the next section, we take use of initial learning rate is 0.01 and cosine decay for finetuning 20 epochs on the ImageNet1000-100 pretrained model.

4. Experiments

In this section, we evaluate dynamic greedy network enlarging method on general image classification dataset ImageNet [6]. We demonstrate the method gets state-of-the-art accuracy with similar MACs.

4.1. Datasets, Networks and Experimental Settings

Dataset. We extensively evaluate our methods on popular classification datasets ImageNet(ILSVRC2012) [6], which contains 1.3M images and 1000 categories, the validation set contains 50K images. On ImageNet, in order to speed up the search process, we create proxy ImageNet1000-100 dataset, which contains 100K train images and 50K validation images randomly sampled from ImageNet train set.

Networks. Two baseline networks are considered: EfficientNet [34] and GhostNet [11]. Original GhostNet architecture follows the setting of MobileNet-v3 [13]. We add squeeze-and-excitation (SE) layer to all blocks. These architectures contain 5 stages according to the spatial sizes of feature maps.

Settings. To accelerate the search process, we set the minimum growth step of resolution and depth as $s_r = 8$ and $s_d = 1$, respectively. For the growth rate of width, we use $s_w = 2$ for small model and $s_w = 4$ for large model. The ratios of MACs between depth and width are in one set $\boldsymbol{P} = \{0.0, 0.1, 0.2, \cdots, 1.0\}$ The error rate of MACs is $\delta = 0.01$. We take use of exponential growth of MACs. We set different number of search iterations $\mathbf{N}$ for small and large models. The finetune method comes from function preserving algorithm [4].

After the process of search is completed, we retrain the acquired network architecture on the whole ImageNet from scratch. The train setting is from timm [38] under its license and EfficientNet [34]. RMSProp optimizer with momentum 0.9; weight decay 1e-5; multi-step learning rate with warmup, initial learning rate 0.064 that decays by 0.97 every 2.4 epochs. Moving average of weight, dropblock [9], random erasing [43] and random augment [5] are used.

ImageNet has noise labels and the method of crop augmentation introduces more noisy input and labels. To prevent this, we use the relabel method [40] to get higher accuracy.

4.2. ImageNet Results and Analysis

Furtherly, the schematic diagram of greedy search for EfficientNet-B1 is shown in Figure 6. Under constrained

Model	Top-1 Acc.	#Params	#MACs	Ratio-to-EfficientNet
EfficientNet-B0 [34]	77.1%	5.3M	0.39B	1x
Ghostnet 1.0× [11]	73.9%	5.2M	0.14B	0.36x
EfficientNet-B1 [34]	79.1%	7.8M	0.69B	1x
ResNet-RS-50 [1]	78.8%	36M	4.6B	6.7x
REGNETY-800MF [26]	76.3%	6.3M	0.8B	1.16x
S-EfficientNet-B1	**79.91%**	8.8M	0.68B	1x
S-EfficientNet-B1-re	**80.71%**	8.8M	0.68B	1x
GhostNet-B1	79.13%	13.3M	0.59B	0.85x
S-GhostNet-B1	**80.08%**	16.2M	0.67B	1x
S-GhostNet-B1-re	**80.87%**	16.2M	0.67B	1x
EfficientNet-B2 [34]	80.1%	9.1M	0.99B	1x
REGNETY-1.6GF [26]	78.0%	11.2M	1.6B	1.6x
S-EfficientNet-B2	**80.92%**	9.3M	1.0B	1x
S-EfficientNet-B2-re	**81.58%**	9.3M	1.0B	1x
EfficientNet-B3 [34]	81.6%	12.2M	1.83B	1x
ResNet-RS-101 [1]	81.2%	64M	12B	6.6x
REGNETY-4.0GF [26]	79.4%	20.6M	4.0B	2.18x
S-EfficientNet-B3	**81.98%**	12.3M	1.88B	1x
S-EfficientNet-B3-re	**82.87%**	12.3M	1.88B	1x
EfficientNet-B4 [34]	82.9%	19.3M	4.39B	1x
REGNETY-8.0GF [26]	79.9%	39.2M	8.0B	1.82x
NFNet-F0 [2]	83.6%	71.5M	12.4B	2.8x
ResNet-RS-152 [1]	83.0%	87M	31B	7.1x
EfficientNetV2-S [35]	83.9%	24M	8.8B	2.0x
S-EfficientNet-B4	**83.0%**	17.0M	4.34B	1x
S-EfficientNet-B4-re	**84.0%**	17.0M	4.34B	1x
GhostNet-B4	82.78%	36.1M	4.39B	1x
S-GhostNet-B4	**83.2%**	32.9M	4.37B	1x
S-GhostNet-B4-re	**84.3%**	32.9M	4.37B	1x

Table 1. Searched Architecture Performance on ImageNet. The '-re' means the models are trained with relabel trick [40]. Our results are in **bold**.

MACs, we show the candidate network architectures. The green box means the best architecture in current iteration, and the gray box are discarded. Besides, the best architecture of each iteration are delivered to the later iterations.

For EfficientNet, we take EfficientNet-B0 as the baseline, and we search the models with MACs similar to EfficientNet-B1 to B4. Besides, we enlarge GhostNet with the principle of EfficientNet and search GhostNet architectures with greedy search method. For GhostNet, we add Squeeze-and-Excitation [15] module for each block. Table.1 shows the main results and comparison with other networks. The searched models are marked with '**S-**'.

GhostNet-B1 and GhostNet-B4 in Table 1 are obtained by the compounding scale rule of EfficientNet. Their performance is lower in contrast to our dynamic greedy search methods. This suggests that the rule on EfficientNet is not fit for GhostNet. We need to resample and optimize for new networks to get suitable rules. Besides, the compounding scale principle ignores the difference of stages, which leads to the loss of elaborate adjustment.

In Table 1, Top-1 accuracies of all searched architectures outperform the compound scaling tricks of EfficientNet [34] and RegNet [26]. On 600M MACs, our searched architectures get 79.91% and 80.08%, improve performance 0.81% and 0.97%, respectively. On EfficientNet-B2 and B3, our searched EfficientNet architectures achieve 80.92% and 81.98%. We search 2 networks on 4B MACs level, S-EfficientNet-B4 gets 83.002% and S-GhostNet-B4 gets 83.23%, respectively.

The relabel training trick improve the accuracy further. The Top-1 accuracy improves 0.6% to 1.1% on all searched architectures. We achieve a new SOTA 80.87% and 84.3% ImageNet top-1 accuracy under the setting of 600M and 4.4B MACs, respectively. All searched network architec-

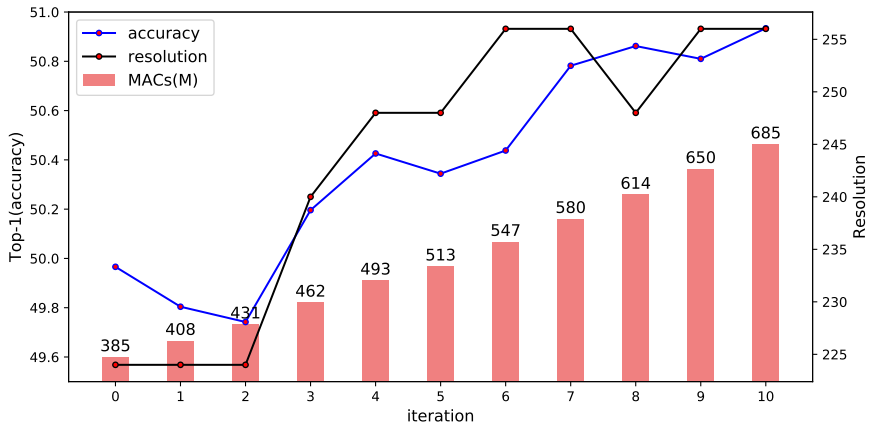

Figure 5. The search process of target 686M MACs of EfficientNet-B1. The blue and black line demonstrate the changes of accuracy and input size as the increase of MACS, respectively.

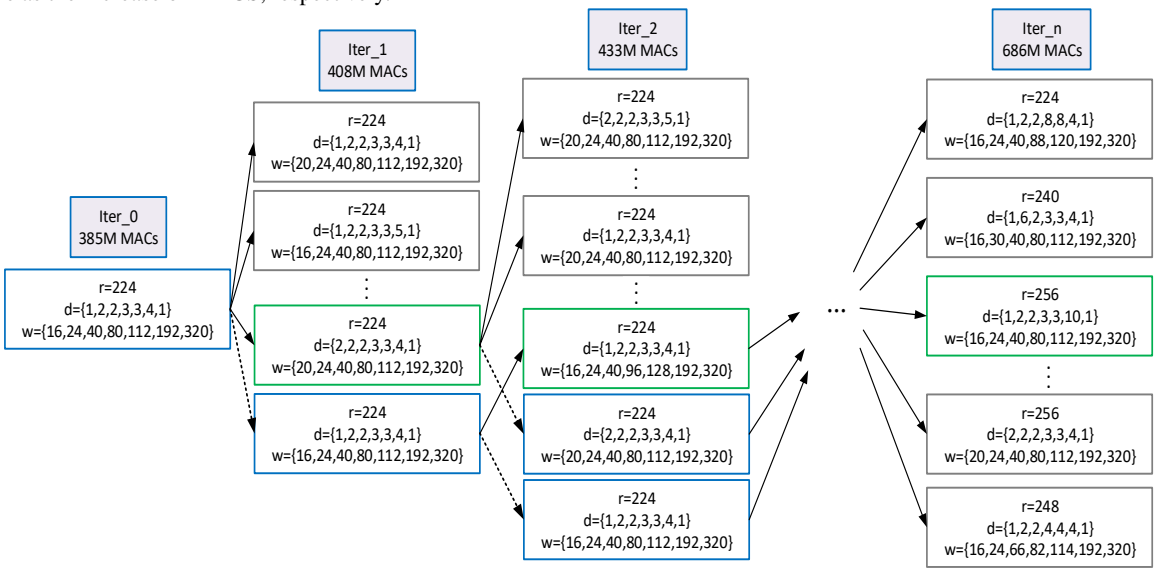

Figure 6. The schematic diagram of greedy search for EfficientNet-B1.

tures are presented in the supplementary materials.

Figure 5 is used specifically to show the changes of accuracy and input resolution of the search process. With increase of MACs, the resolution rises wavily, which verifies the role of dynamic search. The accuracy increases slowly and steadily.

5. Conclusion

Scaling up network architecture is an effective way for generating deep neural networks with excellent performance. Different from the conventional approaches that directly enlarge the given network by using a unified rule, we present a novel dynamic greedy network enlarging algorithm, which maximize the utilization of MACs for all stages. The entire network enlarging task is divided into many iterations for searching the best computational allocation in a step-by-step fashion. In the process of amplifying the base model, the added MACs will be assigned to the most appropriate location. Experimental results on several benchmark models and datasets show that the proposed method is able to surpass the original unified enlarging scheme and achieves state-of-the-art network performance in terms of both network accuracy and computational costs. Beyond allocation of MACs in the stage level, more fine-grained allocation of MACs are expected and more effective and time-saving methods is still needed.

References

[1] Irwan Bello, William Fedus, Xianzhi Du, Ekin D Cubuk, Aravind Srinivas, Tsung-Yi Lin, Jonathon Shlens, and Barret Zoph. Revisiting resnets: Improved training and scaling strategies. *arXiv preprint arXiv:2103.07579*, 2021. 7

[2] Andrew Brock, Soham De, Samuel L Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. *arXiv preprint arXiv:2102.06171*, 2021. 7

[3] Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. Efficient architecture search by network transformation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 5

[4] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015. 5, 6

[5] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020. 6

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 6

[7] Thomas Elsken, Jan-Hendrik Metzen, and Frank Hutter. Simple and efficient architecture search for convolutional neural networks. *arXiv preprint arXiv:1711.04528*, 2017. 5

[8] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey, 2019. 2

[9] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. *arXiv preprint arXiv:1810.12890*, 2018. 6

[10] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7036–7045, 2019. 2

[11] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1580–1589, 2020. 2, 6, 7

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2, 3

[13] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1314–1324, 2019. 2, 6

[14] 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 preprint arXiv:1704.04861*, 2017. 2

[15] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 7

[16] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *Advances in neural information processing systems*, pages 103–112, 2019. 1

[17] Chenhan Jiang, Hang Xu, Wei Zhang, Xiaodan Liang, and Zhenguo Li. Sp-nas: Serial-to-parallel backbone search for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11863–11872, 2020. 2

[18] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016. 1

[19] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pages 528–536, 2017. 5

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2

[21] 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. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 82–92, 2019. 2

[22] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018. 2

[23] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 2

[24] Zechun Liu, Xiangyu Zhang, Zhiqiang Shen, Yichen Wei, Kwang-Ting Cheng, and Jian Sun. Joint multi-dimension pruning via numerical gradient update. *IEEE Transactions on Image Processing*, 30:8034–8045, 2021. 2

[25] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018. 2

[26] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020. 1, 3, 5, 7

[27] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019. 2

[28] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):4780–4789, Jul. 2019. 5

[29] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1

[30] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 2

[31] Albert Shaw, Daniel Hunter, Forrest Landola, and Sammy Sidhu. Squeezenas: Fast neural architecture search for faster semantic segmentation. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 0–0, 2019. 2

[32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 2

[33] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 2

[34] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 1, 2, 3, 6, 7

[35] Mingxing Tan and Quoc V Le. Efficientnetv2: Smaller models and faster training. *arXiv preprint arXiv:2104.00298*, 2021. 7

[36] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10781–10790, 2020. 2

[37] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, et al. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12965–12974, 2020. 2

[38] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 6

[39] Hang Xu, Lewei Yao, Wei Zhang, Xiaodan Liang, and Zhenguo Li. Auto-fpn: Automatic network architecture adaptation for object detection beyond classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6649–6658, 2019. 2

[40] Sangdoo Yun, Seong Joon Oh, Byeongho Heo, Dongyoon Han, Junsuk Choe, and Sanghyuk Chun. Re-labeling imagenet: from single to multi-labels, from global to localized labels. *arXiv preprint arXiv:2101.05022*, 2021. 6, 7

[41] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 1, 2

[42] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018. 2

[43] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13001–13008, 2020. 6

[44] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018. 2, 5