EFFICIENT NET: Exploration on Compound Scaling Method

Members of Group 22
A0236299N, A0236326H, A0251125U, A0236307J, A0251245M
YANG SIZHE, HE YICHEN, SONG MINGYUE, WANG JIANGYI, LOU XINYUN

Abstract

In this project, we conduct a systematic study on scaling convolutional neural networks (ConvNets) to enhance their performance. Our empirical experiments reveal that achieving optimal gains involves a careful balancing of network depth, width, and resolution. Building upon these observations, we propose a novel scaling method that uniformly scales all dimensions of depth, width, and resolution using a straightforward but highly effective compound coefficient. To go even further, we make additional adjustments to our model structure. Our model achieves highest accuracy compared to ConvNet baseline and ResNet-152 among all tasks.

1 Introduction

Since AlexNet (Krizhevsky et al., 2017) won the 2012 ImageNet competition, ConvNets have become increasingly more accurate by going bigger. More sophisticated architectures and larger scales are used for boosting model performance. For example, ResNet (He et al., 2016) can be scaled up from ResNet-18 to ResNet-200 by using more layers. However, the process of scaling up ConvNets has never been well understood and there are currently many ways to do it. Generally speaking, we can scale up ConvNets by adding more layers (scaling up depth), scale up models by input image resolution, and add more channels (scaling up depth). While it is possible to scale up multiple dimensions simultaneously, scaling across different dimensions can be interrelated.

In this project, we use the EfficientNet paper (Tan and Le, 2019) as our guide to study and rethink the process of scaling up ConvNets. We conduct empirical experiments by scaling up only single dimension and multiple dimensions based on three classes CIFAR-10 dataset (Figure 1). Our results suggest that scaling up any single dimension of network improves accuracy, but the degree of benefit diminishes as the models grow larger. However, if we balance all dimensions of network, our model can achieve better performance (Figure 6 in Appendix 9.1). Based on these observations, we propose a simple yet effective compound scaling method. Suppose we want to use 2^N times more computational resources, then we can simply increase the network depth by α^N, width by β^N, and image size by γ^N, where α, β, and γ are constant coefficients determined by a small grid search on the original small model.

In addition, we have identified two key drawbacks associated with the basic structure (MB Convolution layer) of EfficientNet : **(1)** it is slow to train on very large images, and **(2)** depthwise convolutions are inefficient in certain layers. To overcome these limitations, we implement a new structure by replacing the the depthwise 3×3 convolution and expansion 1×1 convolution in MBConv with a regular 3×3 convolution. To differentiate, we refer to the original and new version of the model as EfficientNet-V1 and EfficientNet-V2 respectively.

Finally, we compare the performance of our model with ConvNet baseline and ResNet-152 on CIFAR-10 and CIFAR-100 datasets. EfficientNet-V2 achieves highest accuracy among all tasks.

2 Dataset

In our experiments, we use these three datasets:

- **CIFAR-10 with ten classes**. We have 50,000 images as training data and 10, 000 images as testing data.

- **CIFAR-10 with three classes**. We only select three classes (cat, deer, and dog) from CIFAR-10.We have 15,000 images as training data and 3, 000 images as testing data.

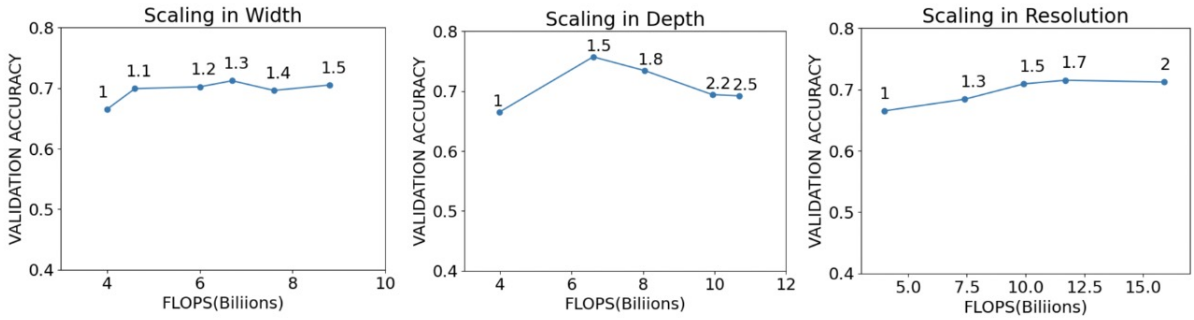

Figure 1: **Scaling Up Baseline Model in Each Dimension.** Generally speaking, bigger networks with larger width, depth and resolution will result in better performance. However, we will suffer from saturation issues. Moreover, increase of depth will lead to degradation of performance due to the dramatic non-linearity of model.

- **CIFAR-100 with one hundred classes**. We have 50,000 images as training data and 10, 000 images as testing data.

3 Motivation

Naively thinking, we could keep adding more layers to pursue a better performance. However, it had been proven that adding more layers can lead to various problems in reality, such as gradient explosion and vanishing gradients. Hence, a creative idea called ResNet (He et al., 2016) network was introduced for its great performance in stabilizing the entire training procedure by incorporating residual connections across different layers.

Another improvement called 'bottleneck' (Howard et al., 2017) was also included. it utilized unique structure with a 1×1 convolutional layer for reducing the number of channels in the input feature maps, followed by a 3×3 filter for extracting spatial features and at last another 1×1 convolutional layer for changing back. The proposition of ResNet has improved the training effectiveness, and most importantly, enabled us to deal with deeper models.

The appearance of ResNet ensured the feasibility of the idea that improving performance could be achieved with adequate computational resources and time. However, in Figure 1, we tried to scale up width (Sandler et al., 2018), depth (He et al., 2016), and resolution (Zoph et al., 2018) respectively, but the model turned out to be saturated. Hence, we required a more wise scaling way instead of naively going deeper in ResNet. To address the challenge, several researchers came up with EfficientNet. Note that in general, EfficientNet utilizes compound scaling up to construct a more efficient structure.

4 Network Architecture Design

In this section, we will discuss the design of network architecture. In Section 4.1, we give mathematical formulation for general ConvNets. In Section 4.2, we decompose network architecture searching into two stages, baseline network searching and compound model scaling. In Section 4.3, we derive the baseline network architecture, which is the starting point of compound model scaling.

4.1 Network Formulation

Generally speaking, for one Convolution Layer i in ConvNet, it can be defined as one function: $Y_i = \mathcal{F}_i(X_i)$. Here, X_i is input tensor, with tensor shape $[H_i, W_i, C_i]$ where H, W stands for spatial dimension and C stands for channel dimension. Similarly, Y_i is output tensor, whose shape is $[H_{i+1}, W_{i+1}, C_{i+1}]$. $\mathcal{F}_i$ is the i-th operator in ConvNet. For the sake of simplicity, we omit the batch size B in the tensor shape. Based on this notation, one ConvNet $\mathcal{N}$ can be formulated as multiple composition of Convolution Layers: $\mathcal{N} = \mathcal{F}_k \odot \cdots \odot \mathcal{F}_2 \odot \mathcal{F}_1(X_1) = \odot_{j=1}^{k} \mathcal{F}_j(X_1)$. This formulation corresponds to one ConvNet with k Convolution Layers. In practice, one ConvNet will be partitioned into several stages. In each stage, all Convolution Layers share the same architecture. Take VGG (Simonyan and Zisserman, 2014) as one example, it has six stages, in which convolution layers share the same convolution type except the for the last pooling layer. Based on this stage design, we can define one ConvNet as:

$$\mathcal{N} = \bigodot_{i=1,\ldots,s} \mathcal{F}_i^{L_i}(X_{[H_i,W_i,C_i]}) \qquad (1)$$

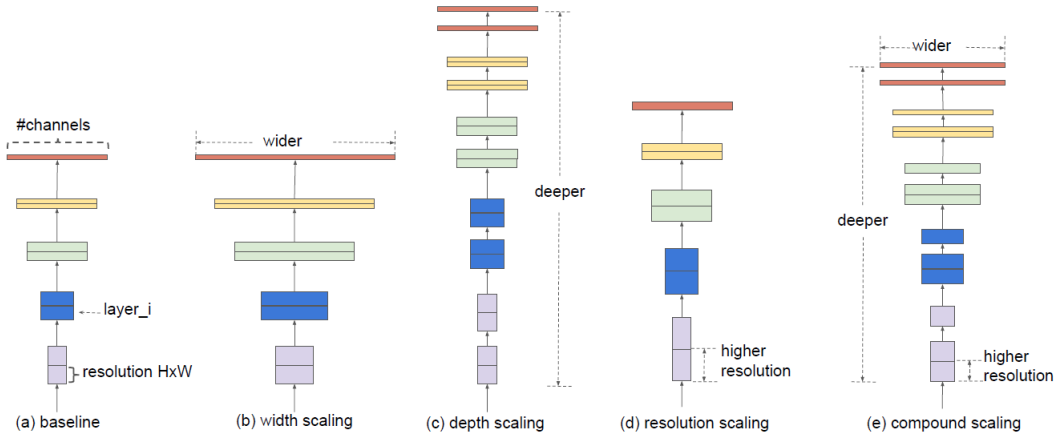

Figure 2: **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

Here, s denotes the number of total stages and $\mathcal{F}_i^{L_i}$ denotes Convolution Layer $\mathcal{F}_i$ is repeated L_i times in i-th stage. $[H_i, W_i, C_i]$ denotes the input tensor shape of i-th stage. To illustrate this, Figure 2(a) gives one example of ConvNet with five stages, that is, we have $s = 5$. Moreover, in each stage, the basic Convolution Layer is repeated twice, which means $L_i = 2$ for $i = 1, 2, ..., 5$. In this example, the spatial dimension (H_i, W_i) is gradually decreased while the channel dimension C_i is increased over layers, which allows the ConvNet to extract features with richer semantic meanings.

4.2 Network Architecture Searching

In order to find the optimal architecture for the network defined in Equation 1, in each stage we are required to find the best convolution operator $\hat{\mathcal{F}}_i$, length of stage $\hat{L}_i$ and spatial and channel dimension $\hat{H}_i, \hat{W}_i, \hat{C}_i$. This approach is computationally intractable due to the complexity of convolution operator. Therefore, instead of searching over the whole space, we attempt to expand L_i, H_i, W_i, C_i without changing the convolution operator $\mathcal{F}_i$. This indeed greatly simplifies the searching problem, which is only related to L_i, H_i, W_i, C_i. However, it remains a large searching space to explore all choices of L_i, H_i, W_i, C_i for each stage.

Therefore, we propose a 2-stage scheme to solve this problem. Firstly, we search for the best tiny baseline network architecture under small range of L_i, H_i, W_i, C_i, which is a tractable problem. The optimal baseline network can be defined as:

$$\mathcal{N}_{base} = \bigodot_{i=1,...,s} \hat{\mathcal{F}}_i^{\hat{L}_i}(X_{[\hat{H}_i, \hat{W}_i, \hat{C}_i]}) \quad (2)$$

Secondly, we use scaling coefficients d, r, w to uniformly expand $\hat{L}_i, \hat{H}_i, \hat{W}_i, \hat{C}_i$. To be more specific, our desired network can be defined as:

$$\mathcal{N}(d, r, w) = \bigodot_{i=1,...,s} \hat{\mathcal{F}}_i^{d \cdot \hat{L}_i}(X_{[r \cdot \hat{H}_i, r \cdot \hat{W}_i, w \cdot \hat{C}_i]}) \quad (3)$$

where w, d, r are coefficients for scaling network width, depth, and resolution. We determine these three coefficients via conducting small grid search followed by uniform scaling up.

4.3 Baseline Architecture Searching

In our 2-stage scheme, we fix the choice of network baseline $\mathcal{N}_{base}$ followed by uniform scaling in all dimensions. Therefore, it is crucial to find one baseline architecture with high quality.

Instead of the manual design, we follow the Network Architecture Searching (Zoph and Le, 2016; Zoph et al., 2018) (NAS) framework, which applies Reinforcement Learning to search for the optimal architecture. We randomly select building blocks of one ConvNet and test its accuracy and computation efficiency, which will play the role as reward. This reward will guide the evolution of our selection criterion.

We develop our baseline network by leveraging a multi-objective searching which takes both accu-

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3×3	224×224	36	1
2	MBConv1,k3×3	112×112	12	1
3	MBConv6,k3×3	112×112	24	2
4	MBConv6,k5×5	56×56	40	2
5	MBConv6,k3×3	28×28	80	3
6	MBConv6,k5×5	14×14	112	3
7	MBConv6,k5×5	14×14	192	4
8	MBConv6,k3×3	7×7	320	1
9	Conv1×1+Pooling+FC	7×7	1280	1

Table 1: **EfficientNet-B0 Model**. The main building blocks are MBConv modules. MBConv6, k3×3 means **(1)** we apply 3×3 convolution operation; and **(2)** bottleneck will enlarge #Channels by 6 times.

racy and FLOPS into consideration. Here, we apply the same searching space as previous work and utilize $ACC(m) \cdot [FLOPS(m)/T]^w$ as our optimizaiton objective. Here, $ACC(m), FLOPS(m)$ denote the accuracy and FLOPS for one specific model m, T is the target FLOPS and w is a hyper-parameter to balance between accuracy and FLOPS. Moreover, we use FLOPS since it is approximately independent with specific hardware. Based on this searching scheme, we achieve one efficient network, which we name as EfficientNet-B0. Table 1 illustrates the detailed architecture of EfficientNet-B0, in which the main building blocks are mobile inverted bottleneck MBConv (Howard et al., 2017) modules.

5 Compound Model Scaling

In this section, we will comprehensively discuss the compound model scaling after deriving EfficientNet-B0 $\mathcal{N}_{base}$. In Section 5.1, we formulate the network searching problem into one optimization problem. In Section 5.2, we introduce compound scaling scheme and its mechanism. In Section 5.3, we propose our general model architecture, which combines baseline architecture EfficientNet-B0 $\mathcal{N}_{base}$ and compound scaling scheme.

5.1 Problem Formulation

As illustrated in Section 4.2, when we derive the best baseline network $\mathcal{N}_{base}$, we are required to find the optimal scaling coefficients d, r, w in Equation 3.Therefore, we can formulate this searching problem into one optimization problem:

$$\begin{aligned} \max_{d,w,r} \quad & Accuracy(\mathcal{N}(d,w,r)) \\ s.t. \quad & \mathcal{N}(d,w,r) = \bigodot_{i=1,\dots,s} \hat{\mathcal{F}}_i^{d\cdot\hat{L}_i}(X_{[r\cdot\hat{H}_i, r\cdot\hat{W}_i, w\cdot\hat{C}_i]}) \\ & \text{Memory}(\mathcal{N}(d,w,r)) \le \text{Memory Limit} \\ & \text{FLOPS}(\mathcal{N}(d,w,r)) \le \text{FLOPS Limit} \end{aligned} \tag{4}$$

Here, $\{\hat{\mathcal{F}}_i, \hat{L}_i, \hat{H}_i, \hat{W}_i, \hat{C}_i\}_{i=1}^{s}$ are predefined by our baseline architecture $\mathcal{N}_{base}$.

5.2 Compound Scaling Scheme

It is worthwhile to mention that, under different resource constraints, we will achieve different optimal $\hat{d}, \hat{w}, \hat{r}$. Moreover, when we have great amount of resources, our searching space will become extremely big, which makes it intractable to solve Optimization Problem 4 directly.

In this report, we propose the **Compound Scaling Scheme**, which applies one compound coefficient ϕ to uniformly scale up width, depth and resolution:

$$\begin{cases} \text{depth: } d &= \alpha^\phi \\ \text{width: } w &= \beta^\phi \\ \text{resolution:} r &= \gamma^\phi \end{cases} \tag{5}$$

where α, β, γ are basic scaling coefficients for depth, width and resolution on small resources. We can solve this small-scale problem via grid search. Based on this design, when we have different resources, we can only adjust the compound coefficient ϕ instead of conducting grid search again and again.

To determine ϕ, notice that, the computational cost for convolution is $O(C_{in} \cdot C_{out} \cdot H_{out} \cdot W_{out})$. Therefore, when we scale up our model by d, w, r, the computational cost is approximately increased by $(d \cdot w^2 \cdot r^2)$ times. Therefore, if we have K times more computational resources, and our grid search for basic scaling coefficients (α, β, γ) is conducted on the constraint $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$, then ϕ should be determined via $\phi = log_2^K$. Under this setting, the computational resources are exactly enlarged by $(\alpha \cdot \beta^2 \cdot \gamma^2)^\phi = 2^\phi = K$ times.

5.3 General Architecture

To conclude, starting from baseline EfficientNet-B0, we can attain more powerful model architecture as the following two steps:

- Step 1: apply grid search to achieve optimal $(\hat{\alpha}, \hat{\beta}, \hat{\gamma})$ for this small-scaled version of Optimization Problem 4:

$$\max_{\alpha,\beta,\gamma} \ Accuracy(\mathcal{N}(\alpha,\beta,\gamma))$$
$$s.t. \ \ \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

- Step 2: assume K times resources are available, then we choose $\hat{\phi} = log_2 K$ and determine the scaling coefficients $\hat{d}, \hat{w}, \hat{r}$ in each dimension via Equation 5.

6 Experiment

To verify the of EfficientNet, we conducted the classification task on CIFAR-10 dataset and compare between three ConvNets, baseline ConvNet, ResNet-152 (He et al., 2016) and EfficientNet-B1.

6.1 Experiment Setting

In each experiment, we followed the same experiment setting. We applied data augmentaion, including randomly rotating, shifting and zooming. We utilized dropout (Srivastava et al., 2014) and batch normalization (Ioffe and Szegedy, 2015) to achieve regularization. Lastly, we used 'Adam' (Kingma and Ba, 2014) as optimizer with default parameters, $\alpha = 0.999, \beta = 0.9$.

6.2 Model Introduction

Baseline ConvNet consists of 3 convolutional layers and 3 fully-connected layers. ResNet-152 comprises 3, 8, 36, 3 residual blocks for the 1^{st}, 2^{nd}, 3^{rd}, 4^{th} stage. We used the pre-trained model on ImageNet (Zeiler and Fergus, 2014). For EfficientNet-B1, we applied the best scaling coefficients ($\alpha = 1.2, \beta = 1.1, \gamma = 1.15$), which are derived in grid search. With this setting, its computational complexity is **approximately 2× larger** than EfficientNet-B0.

6.3 Experiment Results

We trained the models on 50,000 images and validated them on 10,000 images, and the reported accuracy was based on the validation set. Table 2 summarizes the performance of baseline ConvNets, ResNet-152 and EfficientNet-B1, including computational performance and classification accuracy.

Model	Params	FLOPS	Acc.
Baseline ConvNet model	1.46M	39.97MMac	0.856
ResNet-152	58.16M	11.58GMac	0.866
EfficientNet-B1	4.02M	398.1MMac	0.929

Table 2: **Comparisons between ConvNet baseline, ResNet-152 and EfficientNet on CIFAR-10.** We measure model complexity via #parameters and FLOPS, and measure performance via validation accuracy.

Regarding computational costs, Table 2 illustrates that, our EfficientNet-B1 only requires 4.02M parameters and 398.1M FLOPS, which means it is indeed an efficient model. Compared to ResNet-152, our EfficientNet-B1 model is 14.5× smaller in parameters and 29× smaller in FLOPS.

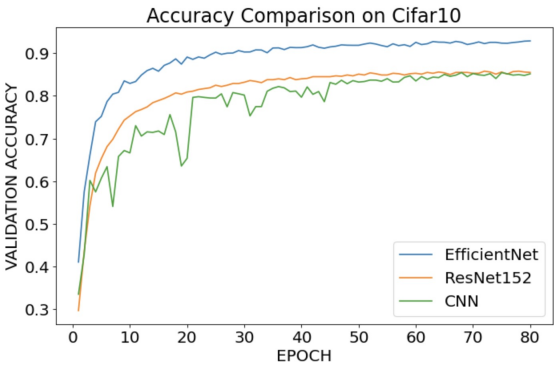

Figure 3: **Accuracy of EfficientNet-B1, ConvNet model and ResNet-152 on CIFAR-10.** EfficientNet achieved better accuracy and stability than Baseline ConvNet Model and ResNet-152.

Regarding accuracy, our EfficientNet-B1 outperformed the other two models. Figure 3 illustrates that, the EfficientNet-B1 achieved higher accuracy and maintained stability throughout the training process.

In conclusion, through the experiment on CIFAR-10, we verified that our EfficientNet-B1 outperformed other models in terms of both computational performance and classification accuracy. This can be attributed to the improvement in network structures and compound scaling technique.

7 Extension

In this section, we extended our work in two directions. Firstly, we trained our model on a larger dataset. Secondly, we compared the performance

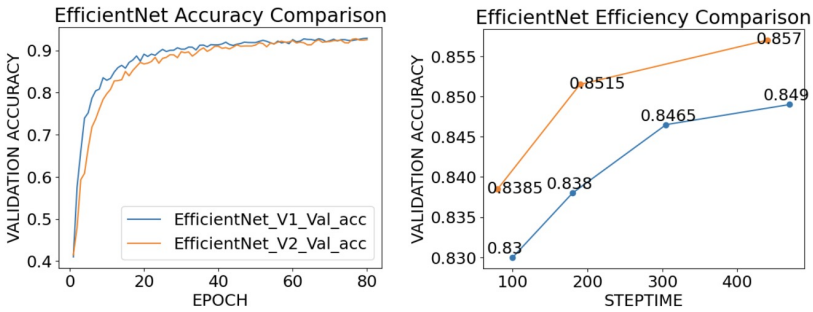

Figure 4: **Comparison between EfficientNet V1 and EfficientNet V2 on CIFAR-10.** EfficientNet V1 performs slightly better than EfficientNet V2, while the former is much more efficient.

of EfficientNet V2 (Tan and Le, 2021), which utilized different convolutional modules.

7.1 Results on Different Dataset

In the first part, we introduced the CIFAR-100 dataset, which consists of 100 classes with 600 images per class. We trained EfficientNet-B1 and a baseline ConvNet on this dataset for 100 epochs using the same hyperparameters and training settings as in previous experiments with CIFAR-10 dataset. Figure 5 showed that EfficientNet-B1 outperformed the ConvNet Baseline greatly, achieving an accuracy of 73.52% compared to less than 50%. This demonstrates the effectiveness of EfficientNet in different image classification scenarios.

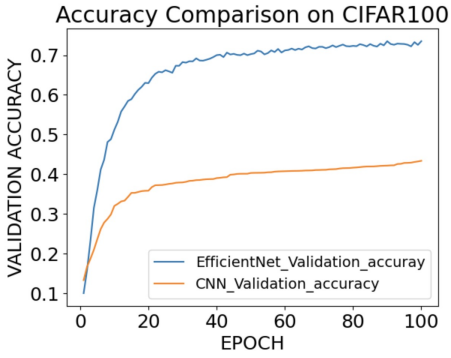

Figure 5: **Validation Accuracy Comparison.** Performance of EfficientNet and ConvNets model on CIFAR-100 dataset. Yellow stands for ConvNet Baseline and blue stands for our EfficientNet.

7.2 Structure Improvement on EfficientNet

Despite its success, the EfficientNet architecture has some limitations. The main limitation is that depthwise convolution operations are slow in the early layers. A new version of EfficientNet (V2) was proposed to release this issue.

EfficientNet V2 replaces MBConv in EfficientNet V1 with Fused_MBConv, which is a regular 3×3 convolution. It allows for faster training with larger image sizes and faster processing with simpler convolutional layers.

To evaluate performance of EfficientNet V2, we trained EfficientNet V2 and V1 on the CIFAR-10 dataset with the same training schedule. Figure 4 showed that EfficientNet V2 achieved similar accuracy compared with V1, while the former was much more efficient in inference time .

In summary, these experiments demonstrate the versatility and effectiveness of EfficientNet in different scenarios. Moreover, EfficientNet V2 releases main limitation of EfficientNet V1 and achieves higher efficiency on the CIFAR-10 dataset. In future work, we could explore the performance of EfficientNet on other datasets and scenarios and compare it with other state-of-the-art models. Additionally, we could explore more on scaling methods and convolutional architecture improvements based on EfficientNet.

8 Conclusion

In this report, we study the scaling of ConvNets in a systematic way and highlight the importance of balancing the network width, depth and resolution. In order to achieve this target, we propose one effective compound scaling method, which enables us to scale up our baseline model to any target resources in a principled way. Based on this method, we propose baseline (EfficientNet-B0) and its scaling-up version (EfficientNet-B1). Through eperiments, it outperforms ResNet-152 with fewer parameters and FLOPS on CIFAR-10 dataset, which illustrates the effectiveness of our proposed EfficientNet-B1.

References

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

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2017. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Mingxing Tan and Quoc Le. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR.

Mingxing Tan and Quoc Le. 2021. Efficientnetv2: Smaller models and faster training. In *International conference on machine learning*, pages 10096–10106. PMLR.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer.

Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.

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

Github Repository Link

Github Repository Link of Group 22

Group Task Assignment

Code Reproduction: Song Mingyue, Wang Jiangyi

Result Reproduction: He Yichen, Lou Xinyun, Yang Sizhe

Slides: He Yichen, Lou Xinyun, Song Mingyue

Representation: Yang Sizhe, Wang Jiangyi

Report Writing: All

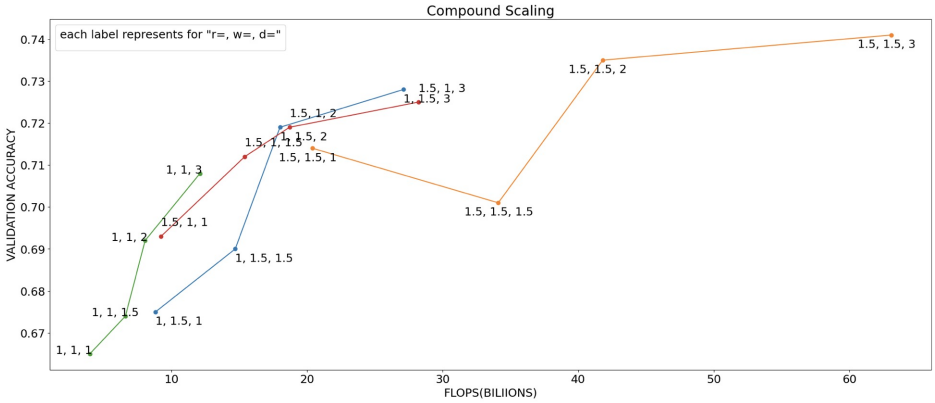

Figure 6: **Compound Scaling on Different Choices of Scaling Coefficients.** Each dot in a line means a different scaling setting. The x-axis is the FLOPS (Floating Point Operation), which is a measure of model complexity. The y-axis is validation accuracy.

9 Appendix

Here is attached some additional details that may provide some helpful context for understanding our works. Since these details are not central to the main report, we put them in this part here for you to check if needed.

9.1 Experiment on Compound Scaling

Figure 6 illustrates the relationship between model performance and scaling coefficients. Different colors represent that, we choose one dimension to scale up from different starting points. For example, green color means, we scale up depth dimension for the starting point $(r = 1, w = 1, d = 1)$. There are two pheonomena can be observed from Figure 6:

- **Phenomenon 1**: Generally speaking, we will achieve better performance while scaling up our models. That is, when we have more FLOPS, we are more likely to achieve better performance in Figure 6.
- **Phenomenon 2**: Scaling up only one dimension will be prone to the saturation of model performance. Figure 6 shows that, the increasing rate of performance slows down when FLOPS is large.

This two phenomena emphasize the importance of our compound scaling method, which balances the size of three scaling coefficients.

9.2 MBConv structure V.S. Fused_MBConv structure

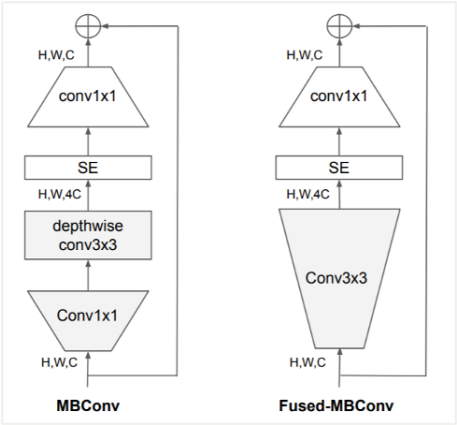

Figure 7: **Structure Comparison.** Structure comparison of MBConv and Fuses_MBConv.

In EfficientNet V1, we applies MBConv Module as shown in Figure 7. This module utilizes 'bottleneck' architecture to reduce computational costs. It firstly uses 1×1 kernels to increase the channels, followed by a 3×3 depthwise kernel to extract local features. Compared with vanilla kernel which is the weighted sum over all channels and pixels, depthwise kernel only sums over pixels within one channel, greatly reducing the computational costs.

In EfficientNet V2, we replace $(1 \times 1$ conv $+ 3 \times 3$ dw-conv$)$ with 3×3 conv only. The main reason is, although the former will be efficient with respect to FLOPS, it does run as fast as we expected in real life due to optimization issues of GPU. The simple 3×3 conv will run faster in real life application.