Artificial Intelligence Research and Development
M. Villaret et al. (Eds.)

doi:10.3233/FAIA210158

Grouped Pointwise Convolutions Significantly Reduces Parameters in EfficientNet

Joao Paulo SCHWARZ SCHULER [a], Santiago ROMANI [a]
Mohamed ABDEL-NASSER [a] Hatem RASHWAN [a] and Domenec PUIG [a]
[a] *Universitat Rovira i Virgili*

Abstract. EfficientNet is a recent Deep Convolutional Neural Network (DCNN) architecture intended to be proportionally extendible in depth, width and resolution. Through its variants, it can achieve state of the art accuracy on the ImageNet classification task as well as on other classical challenges. Although its name refers to its efficiency with respect to the ratio between outcome (accuracy) and needed resources (number of parameters, flops), we are studying a method to reduce the original number of trainable parameters by more than 84% while keeping a very similar degree of accuracy. Our proposal is to improve the pointwise (1x1) convolutions, whose number of parameters rapidly grows due to the multiplication of the number of filters by the number of input channels that come from the previous layer. Basically, our tweak consists in grouping filters into parallel branches, where each branch processes a fraction of the input channels. However, by doing so, the learning capability of the DCNN is degraded. To avoid this effect, we suggest interleaving the output of filters from different branches at intermediate layers of consecutive pointwise convolutions. Our experiments with the CIFAR-10 dataset show that our optimized EfficientNet has similar learning capacity to the original layout when training from scratch.

Keywords. EfficientNet, Deep Learning, Computer Vision, Classification, CNN, DCNN

1. Introduction

Plenty of image classification architectures are tested and benchmarked with ImageNet [1] dataset. On the other hand, it should be noted that not all problems in image classification have 1000 classes and millions of samples nor every research group has the required computing resources to train deep neural network models on large datasets. In this sense, we propose a highly parameter-efficient DCNN architecture that performs well at training from scratch with small datasets and small computing resources. As an example, in the scope of plant disease classification, the Cropped-PlantDoc dataset [2] has less than 10k image samples. Due to its small sample size, this dataset is prone to overfitting. We show that our highly parameter-efficient architecture performs better than the baseline when training them with small datasets.

This article is structured as follows: section 2 presents and discusses relevant work in regards to DCNNs, parameter-efficient DCNNs and datasets used in this work. Section 3 presents our proposed pointwise convolution replacement. Our results and discussion are given in sections 4 and 5, respectively. Section 6 summarizes the paper.

2. Related work

In 1980, Fukushima [3] devised a layered artificial neural network for image classification inspired by the visual cortex structure. In that network, the first layer contains neurons that detect simpler patterns with a small receptive field. Deeper layers detect more complex patterns with wider receptive fields by composing patters from previous layers. That was the first Convolutional Neural Network (CNN).

In 2012, Krizhevsky et al. [4] reported a major breakthrough in the ImageNet Large Scale Visual Recognition Challenge, using their AlexNet architecture. Since then, many other DCNN architectures have been introduced, like ZFNet [5], VGG [6], GoogLeNet [7], ResNet [8] and DenseNet [9]. Since the number of layers of proposed convolutional neural networks have increased from 5 to more than 200, those models are usually referred as Deep Learning or DCNN.

In regards to datasets, the following three are in our interest:

- The Oxford-IIIT Pet dataset [10] consists of 25 breeds (classes) of dogs and 12 breeds (classes) of cats. In total, there are 37 classes of images. Each class has around 200 images. Images have various sizes and complex backgrounds and illumination patterns.
- CIFAR-10 dataset [11] consists of 60k 32x32 images belonging to 10 different classes: airplane, automobile, bird, cat, deer, dog, frog, horse ship and truck. These images are taken from natural and uncontrolled lightning environment. They contain only one prominent instance of the object to which the class refers to. The object may be partially occluded or seen from an unusual viewpoint.
- Cropped-PlantDoc dataset [2] was devised to allow plant leaf disease classification. It was created by cropping individual leafs from a smaller dataset called PlantDoc that contained multiple leafs per image. This dataset has 13 plant species and 27 classes for different diseases on each specie. Images have complex backgrounds and the area covered by the leafs has varying sizes.

These datasets offer together an interesting broad set of classes. Cropped-PlantDoc contains plants only. The Oxford-IIIT Pet dataset contains animals only. The CIFAR-10 dataset contains animals and man made objects. Besides, they are relatively small and allow easy replication of our ideas with affordable hardware and small computing time.

To be able to reduce the number of weights in our DCNNs, we propose the use of grouped convolutions. A grouped convolution evenly separates input channels and neurons for each group. Each neuron processes only input channels entering its own group. This drastically reduces the number of weights and floating point computations. A depthwise convolution is an extreme case which each group has only one input channel and only one filter. In a depthwise convolution, the number of input channels, filters and groups are the same. In AlexNet, due to implementation constraints, convolutions were separated into two groups. In 2016, Ioannou at al. [12] experimented grouped convolutions with 2, 4, 8 and 16 groups per convolution for CIFAR-10 classification. Ioannou at al. showed that replacing 3x3 and 5x5 common convolutions by grouped convolutions can reduce the number of parameters by more than 50%. They also showed that their split architectures can keep the original classification accuracy or even improve it slightly. In their work, there was no attempt to optimize the 1x1 pointwise convolutions, i.e., convolutions that have 1x1 kernels with one trainable parameter per input channel. These kernels do not take into account neighboring positions such as, for example, 3x3 filters.

In 2017, Howard at al. [13] developed an architecture called MobileNet. The MobileNet building block is composed by a depthwise separable convolution followed by a pointwise convolution. MobileNets are parameter-efficient when compared to previous models. As an example, MobileNet-160 has nearly 45 times less parameters than AlexNet and achieves similar accuracy when classifying the ImageNet dataset. MobileNet-224 has nearly 40% less parameters than GoogLeNet and achieves higher accuracy than GoogLeNet. Howard at al. noted that as their models are more parameter-efficient, these smaller models require less data augmentation. There is an aspect in their MobileNet models that has central interest for our proposal: nearly 75% of the parameters and 95% of multiplications and additions are computed by pointwise convolutions. This makes a strong case for an optimized pointwise convolution.

Also in 2017, Ting Zhang et. al. [14] proposed mixing grouped convolutions with interleaving layers. They proposed a grouped spatial convolution followed by an interleaving layer and a grouped pointwise convolution. The most evident difference from their work to ours is about us developing a solution specifically targeting a pointwise convolution replacement.

In 2019, Mingxing Tan et al. [15] developed the EfficientNet architecture. At that time, their EfficientNet-B7 variant was 8.4 times more parameter-efficient and 6.1 times faster than the best existing architecture achieving 84.3% top-1 accuracy on ImageNet. As in the case of MobileNets, more than 80% of the parameters of EfficientNets come from standard pointwise convolutions. This aspect opens an opportunity for a huge reduction in number of parameters and floating point operations.

3. Methodology

Latest DCNN architectures have a big portion of their parameters located in pointwise convolutions. Thus, we propose replacing pointwise convolutions by parameter-efficient counterparts. Figure 1 shows a diagram of our proposed pointwise replacement. It starts with a pointwise grouped convolution K (parallel groups K_1 to K_{N_i}) followed by a channel interleaving layer which mixes channels for the next pointwise grouped convolution L (parallel groups L_1 to L_{N_i}). All channels from parallel groups K_1 to K_{N_i} are concatenated into a single output of the K layer. The same happens for the L layer. Concatenated outputs from K and L layers are summed channel by channel, which makes the L layer to behave as a residual convolution.

In standard pointwise convolutions, each filter has one trainable parameter per input channel. Therefore, the number of parameters P in layer i is calculated from the number of channels of the preceding activation map C_{i-1} and the number of filters F_i as in Eq. 1:

$$P_i = C_{i-1} \cdot F_i \tag{1}$$

For grouped convolutions, let us note the number of groups in layer i as N_i. Each group is fed a contiguous subset of C_{i-1}/N_i of the input channels. Moreover, the number of filters per group is F_i/N_i. Thus, the number of parameters per group is the number of filters per group multiplied by the number of channels per group $(F_i/N_i) \cdot (C_{i-1}/N_i)$. Therefore, multiplying the previous expression by the number of groups, we obtain the total number of parameters of a grouped convolutional layer as in Eq. 2:

$$P_i = (C_{i-1} \cdot F_i)/N_i \tag{2}$$

When grouping convolutions, we follow these constraints:

- Each group must have at least 16 input channels. This will allow a minimum degree of intragroup combinations.
- The number of groups N_i shall be the greatest common divisor of the number of input channels C_{i-1} and the number of filters F_i respecting the previous constraint ($C_{i-1}/N_i \geq 16$).
- Given above constraints, if we cannot have more than one group, then the original pointwise convolution is not replaced by this sub-architecture.
- Only when the number of output channels per group (F_i/N_i) is bigger than 1, an interleaving layer will be added.
- Only when the number of input channels is greater or equal than the number of output channels ($C_{i-1} \geq C_i$), a grouped pointwise convolutional layer is added after the interleaving layer and then the result of both grouped convolutional layers are summed.

As an example, in a monolithic pointwise convolution with $C_{i-1} = 1,024$ and $F_i = 512$, we will obtain $P_i = 524,288$ parameters. If we replace this pointwise convolution with our sub-architecture, according to the constraint of a minimum of 16 channels per group, N_i must be 64. In this example, the first grouped convolutional layer will have $1,024 \cdot 512/64 = 8,192$ parameters. The second grouped convolutional layer will also have 64 groups, but the number of input channels will be 512, hence the total number of parameters will be $512 \cdot 512/64 = 4,096$. Summing both results, the total number of parameter of the whole sub-architecture will be 12,288, which is a saving of almost 97.7% from the original parameter count.

In all of our experiments, we used an Amazon AWS instance with a single Tesla T4 GPU paired with 8 virtual cores. In regards to software, we used Keras/Tensorflow and RMSProp optimizer. Our experiment with Oxford-IIIT Pet dataset has cyclical learning rate of 25 epochs. All training experiments have data augmentation. For each dataset, we used a specific number of epochs as shown in Table 1. We used more epochs in smaller datasets to compensate the smaller number of samples. The number of epochs fits a multiple of our learning rate cycle. In this work, we did not use transfer learning as our goal is to evaluate learning capacity of parameter-efficient models. In all experiments, we kept the same dropout as per original EfficientNet (baseline) implementation.

Our source code is publicly available and can be fount at `https://github.com/joaopauloschuler/kEffNet/` .

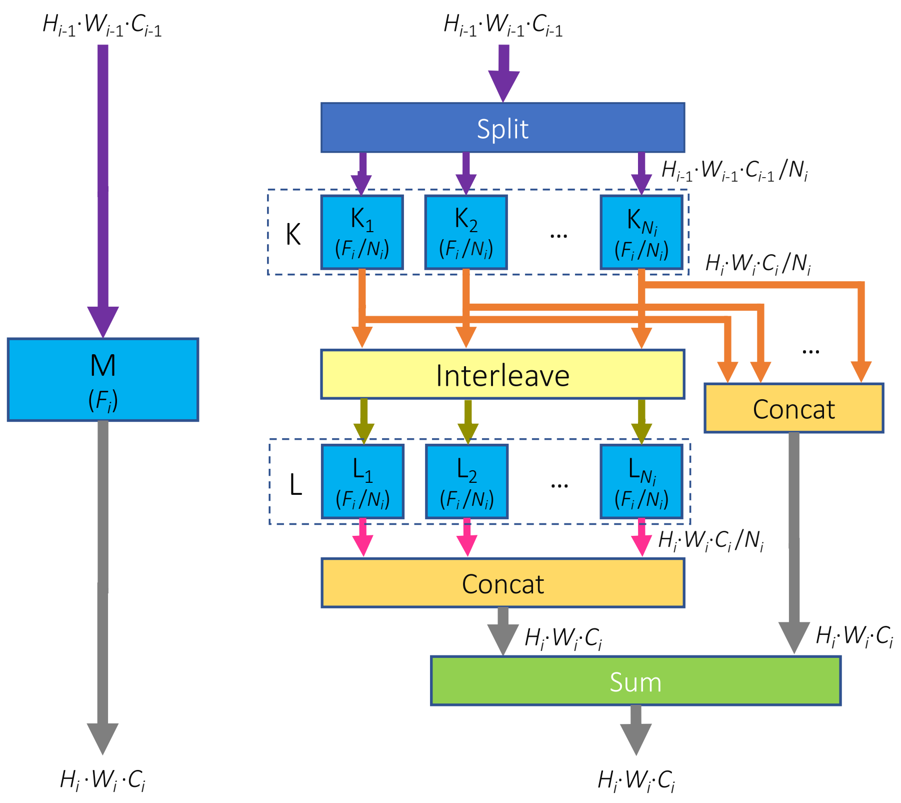
$H_{i-1} \cdot W_{i-1} \cdot C_{i-1}$
M
(F_i)
$H_i \cdot W_i \cdot C_i$
$H_{i-1} \cdot W_{i-1} \cdot C_{i-1}$
Split
$H_{i-1} \cdot W_{i-1} \cdot C_{i-1} / N_i$
K
K_1 (F_i/N_i)
K_2 (F_i/N_i)
...
K_{N_i} (F_i/N_i)
$H_i \cdot W_i \cdot C_i / N_i$
Interleave
...
Concat
L
L_1 (F_i/N_i)
L_2 (F_i/N_i)
...
L_{N_i} (F_i/N_i)
$H_i \cdot W_i \cdot C_i / N_i$
Concat
$H_i \cdot W_i \cdot C_i$
$H_i \cdot W_i \cdot C_i$
Sum
$H_i \cdot W_i \cdot C_i$

Figure 1. Graphical representation of our pointwise convolution replacement. At the left, a classic monolithic layer M with F_i pointwise filters. At the right, our substitute for M, made of two grouped pointwise convolutional layers, K and L, with N_i parallel groups in each layer. Each parallel group has F_i/N_i filters. The size of activation maps, transmitted through arrows, is the multiplication of height H, width W and number of channels C, sometimes divided by N_i. The subindex i indicates architecture level. For pointwise convolutions, H_i=H_{i-1}, W_i=W_{i-1} and C_i=F_i.

4. Results

We have analyzed two types of results: test classification accuracy and heatmaps. In the following subsections, we name our modified EfficientNet variants as ”kEffNet” followed by the corresponding complexity code ”-Bn” (”-B0”, ”-B1”, etc.).

4.1. Test Classification Accuracy

Table 1 compares test accuracies and trainable parameter count of the baseline EfficientNet-B0 and our variant kEffNet-B0 with the tested datasets.

As the scope of this work is limited to small datasets and small architectures, we only experimented with the smallest EfficientNet variant (EfficientNet-B0) and our modified variant (kEffNet-B0). Nevertheless, Table 2 provides the number of trainable parameters of the other EfficientNet variants (original and modified).

dataset	classes	epochs	model	parameters	test accuracy
Cropped-PlantDoc	29	75	EfficientNet-B0	4,044,697	49.08%
			kEffNet-B0	664,041	**64.39%**
Oxford-IIIT Pet	37	150	EfficientNet-B0	4,054,945	56.73%
			kEffNet-B0	674,289	**60.09%**
CIFAR-10	10	50	EfficientNet-B0	4,020,358	**91.66%**
			kEffNet-B0	639,702	91.64%

Table 1. Classification accuracies found with baseline and our variant.

variant	EfficientNet Parameters	kEffNet Parameters	Reduction
B0	4,020,358	639,702	84.09%
B1	6,525,994	922,770	85.86%
B2	7,715,084	1,130,344	85.35%
B3	10,711,602	1,532,902	85.69%
B4	17,566,546	1,952,306	88.89%
B5	28,361,274	2,633,470	90.71%
B6	40,758,754	4,714,030	88.43%
B7	63,812,570	4,669,218	92.68%

Table 2. Number of trainable parameters per EfficientNet and kEffNet variants, for a 10 classes dataset.

4.2. Heatmap

We produced heatmaps for a number of samples from the Oxford-IIIT Pet as shown in Figure 2. We observed that our kEffNet-B0 tends to concentrate feature activation close to the ears and to the top of the cats head, while the baseline frequently finds feature activation in the background.

In Figure 3, we find an image sample that both the baseline and our model do not focus on the cat. We speculate that the proposed model is detecting textile patterns that are commonly found in cat photos.

5. Discussion

We face two main limitations when applying grouped convolutions:

1. It prevents the DCNN from exploring all possible combinations among all features coming from the previous layer due to missing connections.
2. The output of parallel groups must be somehow joined.

To alleviate the first and partially the second limitation, we interleave the channels of the first grouped pointwise convolution before feeding the next grouped convolution. To alleviate the second limitation, we propose the use of summation operator for joining paths. Compared to concatenation, summation has the advantage of not increasing the number of output channels. Our pointwise parameter-efficient replacement does not directly explain why we got higher classification accuracy in the Cropped-PlantDoc dataset or in the Oxford-IIIT Pet dataset. Most likely, as we have less trainable parameters, our modified kEffNets are less prone to overfitting. As CIFAR-10 is comparatively a bigger dataset, we found almost the same test classification accuracy in baseline and modified

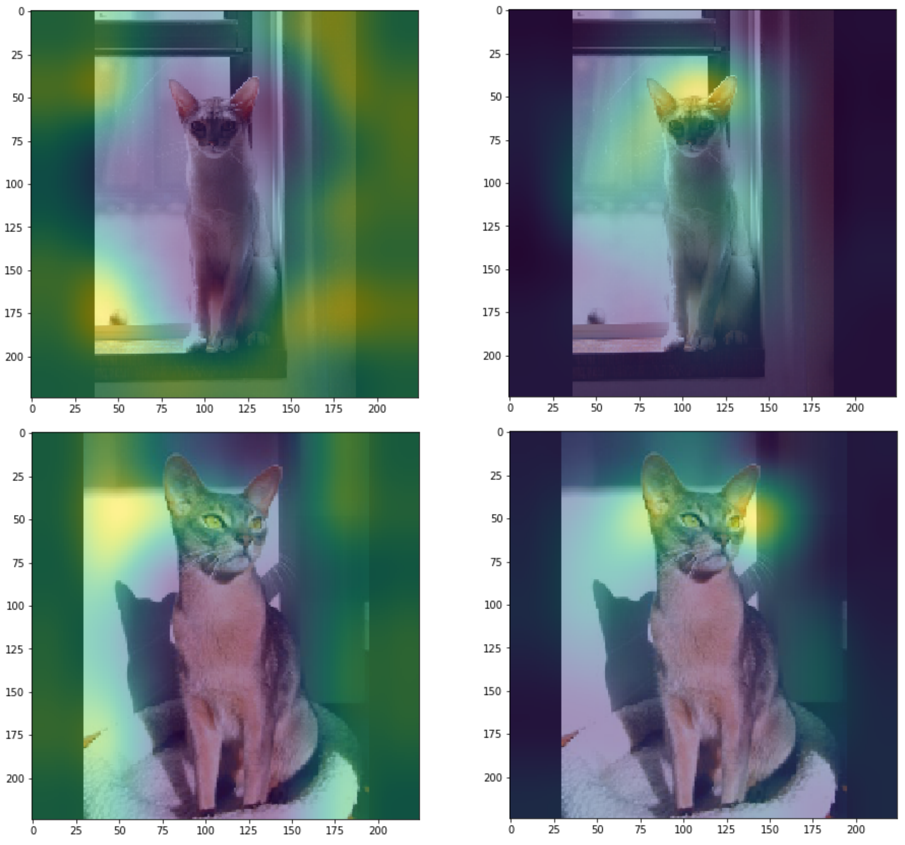

Figure 2. On the left, heatmaps produced by the EfficientNet-B0 baseline. On the right, heatmaps produced with our kEffNet-B0.

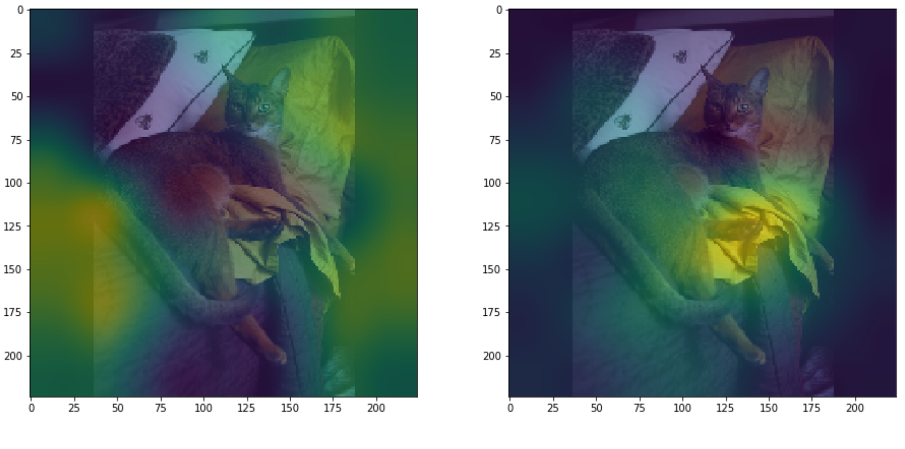

Figure 3. On the left, heatmap produced by the EfficientNet-B0 baseline. On the right, heatmap produced with our kEffNet-B0.

kEffNet. In regards to heatmaps, the baseline seems to be more frequently concentrating attention on the background. This effect may not be necessarily a product of overfitting.

6. Conclusion

In this work, we have experimented with a replacement of EfficientNet pointwise convolutions using a sub-architecture that contains up to two grouped convolutions, an interleaving layer and a summation layer at its end. By doing this replacement in the EfficientNet-B0 variant, we were able to save more than 84% of the trainable parameters while keeping the classification accuracy on the CIFAR-10 dataset when training from scratch. As we have less trainable parameters, we found significantly better classification accuracy in the Cropped-PlantDoc (+15.3%) and Oxford-IIIT Pet datasets (+3.3%) probably due to less overfitting. As a matter of future research, we may delve in fine tuning our sub-architecture details such as decreasing the dropout rate as we having less trainable parameters.

References

[1] Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, et al. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV). 2015;115(3):211-52.

[2] Singh D, Jain N, Jain P, Kayal P, Kumawat S, Batra N. PlantDoc: A Dataset for Visual Plant Disease Detection. CoRR. 2019;abs/1911.10317. Available from: http://arxiv.org/abs/1911.10317.

[3] Fukushima K. Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. Biological Cybernetics. 1980;36:193-202.

[4] Krizhevsky A, Sutskever I, Hinton GE. ImageNet Classification with Deep Convolutional Neural Networks. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ, editors. Advances in Neural Information Processing Systems 25. Curran Associates, Inc.; 2012. p. 1097-105. Available from: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[5] Zeiler MD, Fergus R. Visualizing and Understanding Convolutional Networks. In: Fleet D, Pajdla T, Schiele B, Tuytelaars T, editors. Computer Vision – ECCV 2014. Cham: Springer International Publishing; 2014. p. 818-33.

[6] Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In: Bengio Y, LeCun Y, editors. 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings; 2015. Available from: http://arxiv.org/abs/1409.1556.

[7] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, et al. Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2015. p. 1-9.

[8] He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition; 2015. Cite arxiv:1512.03385Comment: Tech report. Available from: http://arxiv.org/abs/1512.03385.

[9] Huang G, Liu Z, Weinberger KQ. Densely Connected Convolutional Networks. CoRR. 2016;abs/1608.06993. Available from: http://arxiv.org/abs/1608.06993.

[10] Parkhi OM, Vedaldi A, Zisserman A, Jawahar CV. Cats and Dogs. In: IEEE Conference on Computer Vision and Pattern Recognition; 2012. .

[11] Krizhevsky A. Learning multiple layers of features from tiny images; 2009.

[12] Ioannou Y, Robertson DP, Cipolla R, Criminisi A. Deep Roots: Improving CNN Efficiency with Hierarchical Filter Groups. CoRR. 2016;abs/1605.06489. Available from: http://arxiv.org/abs/1605.06489.

[13] Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. CoRR. 2017;abs/1704.04861. Available from: http://arxiv.org/abs/1704.04861.

[14] Zhang T, Qi G, Xiao B, Wang J. Interleaved Group Convolutions for Deep Neural Networks. CoRR. 2017;abs/1707.02725. Available from: `http://arxiv.org/abs/1707.02725`.

[15] Tan M, Le QV. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. CoRR. 2019;abs/1905.11946. Available from: `http://arxiv.org/abs/1905.11946`.