ISSN: 1803-3814 (Printed), 2571-3701 (Online)
https://doi.org/10.13164/mendel.2022.1.023

Grouped Pointwise Convolutions Reduce Parameters in Convolutional Neural Networks

Joao Paulo Schwarz Schuler[1,✉] **Santiago Romani**[1],**Mohamed Abdel-Nasser**[1,2], **Hatem Rashwan**[1], **Domenec Puig**[1]

[1]DEIM, Universitat Rovira i Virgili, Spain
[2]Electrical Engineering Department, Aswan University, Aswan, Egypt

joaopaulo.schwarz@estudiants.urv.cat[✉], santiago.romani@urv.cat, mohamed.abdelnasser@urv.cat, hatem.abdellatif@urv.cat, domenec.puig@urv.cat

Abstract

In Deep Convolutional Neural Networks (DCNNs), the parameter count in pointwise convolutions quickly grows due to the multiplication of the filters and input channels from the preceding layer. To handle this growth, we propose a new technique that makes pointwise convolutions parameter-efficient via employing parallel branching, where each branch contains a group of filters and processes a fraction of the input channels. To avoid degrading the learning capability of DCNNs, we propose interleaving the filters' output from separate branches at intermediate layers of successive pointwise convolutions. To demonstrate the efficacy of the proposed technique, we apply it to various state-of-the-art DCNNs, namely EfficientNet, DenseNet-BC L100, MobileNet and MobileNet V3 Large. The performance of these DCNNs with and without the proposed method is compared on CIFAR-10, CIFAR-100, Cropped-PlantDoc and Oxford-IIIT Pet datasets. The experimental results demonstrated that DCNNs with the proposed technique, when trained from scratch, obtained similar test accuracies to the original EfficientNet and MobileNet V3 Large architectures while saving up to 90% of the parameters and 63% of the floating-point computations.

Keywords: *Deep Learning, Deep Convolutional Neural Networks, Interleaving, Grouped Convolution, Image Classification.*

Received: 01 February 2022
Accepted: 26 April 2022
Online: 27 April 2022
Published: 30 June 2022

1 Introduction

A grouped convolution in DCNNs divides input channels and filters into groups. Each group of filters can be understood as an independent (parallel) path for information to flow. Instead of processing all input channels, in a grouped convolution, each filter processes only input channels belonging to the same group. This grouping reduces the number of weights in each filter and consequently the number of floating-point computations. Notably, 1x1 filters with one trainable parameter per input channel compose pointwise convolution. Unlike (spatial) 3x3 convolutional filters, these filters do not consider surrounding positions.

This paper proposes an efficient method to optimize any DCNN architecture by grouping pointwise convolutions found in its original design. Besides, we propose interleaving the filters' output from separate groups at intermediate levels of successive pointwise convolutions to prevent diminishing the learning power of DCNNs. The resulting architecture is highly parameter-efficient and performs well at training from scratch with datasets that contain few image samples. This architecture also requires less floating point operations. For instance, in the context of plant disease classification, the Cropped-PlantDoc dataset [16] contains less than 10 thousand images. It should be noted that the Cropped-PlantDoc dataset is prone to overfitting when using classical heavy DCNN architectures as a consequence of the small sample count. We demonstrate that the proposed pointwise convolution optimization can significantly reduce the number of parameters of DCNNs while performing better than the baseline models when training them with low sample count datasets. Our previous study in [13] showed that the pointwise convolution optimization technique could enhance the efficiency of EfficientNets. The present study demonstrates that the proposed pointwise convolution optimization technique can be applied to most state-of-the-art DCNN architectures. It is worth noting that achieving the state-of-the-art classification accuracy on any image classification dataset is out of the scope of this paper, as we focus on DCNN optimization.

This is how this article is organized: Section 2 introduces and examines relevant work on DCNNs, parameter-efficient DCNNs, and the datasets used in this study. The proposed pointwise convolution optimization is explained in Section 3. Sections 4 and 5 provide the experimental results, comparisons and discussion, respectively. The paper is summarized in Section 6.

2 Related work

One decade ago, Krizhevsky et al. [9] proposed the AlexNet architecture that achieved a breakthrough in the ImageNet Large Scale Visual Recognition Challenge. AlexNet architecture contains 5 convolutional layers and 3 dense layers. Since then, numerous DCNN architectures, such as ZFNet [20], VGG [15], GoogLeNet [17], ResNet [3] and DenseNet [6] have been presented. Such models are commonly called "Deep Learning" or DCNN as the number of convolutional layers has expanded from 5 to more than 200.

In 2013, Min Lin et al. introduced the Network in Network architecture (NiN)[10]. This architecture contains 3 spatial convolutional layers with 192 filters each, interspersed with pairs of pointwise convolutional layers. The pairs of pointwise convolutions allow the network to learn complex patterns without the computational cost of a spatial convolution.

They were able to attain state-of-the-art classification accuracies in the CIFAR-10 and CIFAR-100 datasets[8] when their work was released.

In 2016, ResNet was introduced. Resnet [3] stacks up to 152 layers with similar topology. Inspired on VGG [15], all ResNet spatial convolutions have 3x3 filters. The authors of ResNet conjectured that deeper CNNs have exponentially low convergence rates. To tackle this problem, they propose skip connections every 2 convolutional layers.

In 2017, Ioannou et al. [7] tested grouped convolutions with various groups per layer (2, 4, 8, and 16) with the image classification using the CIFAR-10 dataset. Working towards optimization for the NiN architecture, Ioannou et al. showed that grouping 3x3 and 5x5 spatial convolutions could decrease the number of parameters by more than 50% when optimizing the NiN architecture. They also demonstrated that their proposed architectures, divided into numerous paths (i.e., groups), may maintain or slightly increase the classification accuracy. Of note, there was no attempt to separate the 1x1 pointwise convolutional layers in their study. Ioannou at. also worked on an optimized ResNet-50 variant by replacing the original spatial convolutions by up to 64 parallel groups. This reduces the number of parameters by 27% and the number of floating-point operations by 37%, while keeping similar classification accuracy on ImageNet dataset. They observed that it is unlikely that every filter depends on the all output channels coming from the previous layer. This observation is very fundamental to our work, as we use it support the idea that grouped convolutions can be as effective as non-grouped filters connected to all incoming channels.

Also in 2017, an improvement for ResNet called ResNeXt[19] was introduced. However, ResNeXt replaces the spatial convolutions by parallel paths, thus reducing the number of parameters. When ResNeXt variants are configured to a similar number of parameters to their original ResNet architectures, ResNeXt variants achieve higher classification accuracy on the ImageNet [12] dataset. In the ResNeXt architecture, the building blocks follow a split-transform-merge paradigm. Although the transforming step is done via parallel spatial convolutions, the splitting and the merging are done by standard (ungrouped) pointwise convolutions.

Howard et al. [5] proposed an architecture called MobileNet. The depthwise separable convolution is an essential component of MobileNet. A depthwise convolution precedes a pointwise convolution in this building block. In comparison to prior models, MobileNets are parameter efficient. MobileNet-160, for instance, has roughly 45 times fewer parameters than AlexNet but achieves equal classification performance when using the ImageNet dataset. MobileNet-224 has roughly 40% fewer parameters than GoogLeNet yet obtained greater accuracy. According to Howard et al., their smaller models need minor data augmentation. An observation of MobileNet models that is crucial to our approach is that pointwise convolutions account for almost 75% of the parameters and 95% of floating-point computations. This architecture is an excellent candidate for our proposal as our proposal saves parameters and computations along pointwise convolutions. Later, in 2019, [4] an improved version of MobileNet was introduced, named as V3, still based on depthwise and pointwise convolutions, but pointwise convolutions remained not grouped.

Zhang et al. [21] mixed grouped convolutions with interleaving layers. Specifically, they proposed a grouped spatial convolution followed by an interleaving layer and a grouped pointwise convolution. The main difference between [21] and our technique is that we target replacing pointwise convolutions.

In [18], Tan et al. proposed the EfficientNet architecture. Their EfficientNet-B7 model was 8.4 times more parameter-efficient and 6.1 times faster than the best architecture at the time, with an ImageNet top-1 accuracy of 84.3%. More than 80% of the parameters in EfficientNets come from standard pointwise convolutions, as well as in MobileNets, which allows for a significant decrease in the number of parameters and floating-point operations, which we have taken advantage of in this study.

It should be noted that the following four image classification datasets are considered in this study:

- The **Oxford-IIIT Pet dataset** [11]: it includes images of 25 breeds of dogs (i.e., 25 classes) and 12 breeds of cats (i.e., 12 classes). There are a total of 37 image classes. There are approximately 200 images in each class. Of note, all images come in various sizes and contain intricate backgrounds and lighting patterns.
- The **CIFAR-10 dataset** [8]: it has 60 thousand 32x32 images of 10 classes (dog, airplane, truck, cat, automobile, bird, horse, deer, frog, and ship). These images were captured in a natural, uncontrolled lighting condition. They only have one visible instance of the object that the class refers to.

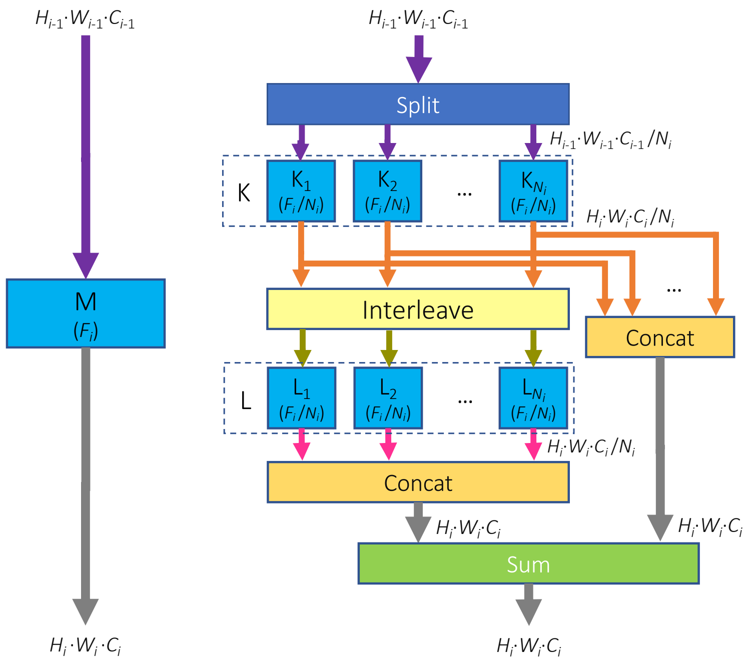

Figure 1: Diagram of the proposed pointwise convolution optimization. (Left) a classic monolithic layer M with F_i pointwise filters. (Right) our proposed replacement for M. It comprises two grouped pointwise convolutional layers (K and L) with N_i groups, where each group consists of F_i/N_i filters. H, W, and C represent the height, width, and number of channels. The size of the activation maps (represented by arrows) equals $H \times W \times C$. In some cases, the activation map size is divided by the number of groups N_i. The i subindex stands for the layer depth. In the case of pointwise convolutions, H_i=H_{i-1}, W_i=W_{i-1} and C_i=F_i.

The objects are sometimes partially obscured or viewed from an unexpected angle.

- The **CIFAR-100 dataset**[8]: it is similar to the CIFAR-10 dataset, but it has 100 classes instead of 10. It contains 60 thousand 32x32 images representing the 100 classes (e.g., machines, plants, animals, and people).
- The **Cropped-PlantDoc dataset** [16]: it was developed for conducting research on plant leaf disease classification. It was formed by cropping individual leaves from the PlantDoc dataset that includes multiple leaves per image. The Cropped-PlantDoc dataset contains 13 plant species and 27 classes. Notably, in this dataset, images have heterogeneous backgrounds and the leaves significantly differ in sizes.

Together, the above 4 datasets bring a variety of object classes that help assess the efficacy of the proposed technique. They are compact datasets and enable easy replication of our proposal using low-cost technology and little computation time.

Some of our experiments output class activation maps (CAMs) [22]. The class activation map method finds image regions used by a CNN to classify an image. Regions that are relevant for the classification are shown from red (more relevant) to blue (less relevant). This method can be used when the last 2 layers of a CNN are a global average pooling and a dense layer. The CAM is calculated from the activation maps preceding the global average pooling and the weights related to the activated image class (filter).

3 Methodology

More than 80% of the parameters in the most recent DCNN architectures are found in pointwise convolutions. Each filter in typical pointwise convolutions has one trainable parameter per input channel. Accordingly, the parameters count P in layer i is computed from the channel count of the prior activation map C_{i-1} and the filters count F_i as expressed in Eq. (1), where F_i is the total number of filters as found in the original monolithic pointwise convolution layer:

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

We propose a method to make pointwise convolutions parameter-efficient. Figure 1 presents the architecture of our proposed optimization. This architecture starts with a pointwise grouped convolution layer K (composed by filter groups K_1 to K_{N_i}) followed by a channel interleaving layer that fuses channels for the subsequent pointwise grouped convolution layer L (filter groups L_1 to L_{N_i}). At the right of our architecture, all channels from groups K_1 to K_{N_i} are concatenated into one path. It is worth noting that the same process occurs for the L layer. The K and L layers' concatenated outputs are summed channel by channel, making the L layer act as a residual convolution.

Indeed, grouped convolutions inherently face a limitation: each parallel group of filters computes its output from their own set of input channels, preventing channels connected to different groups to be combined with each other. To alleviate this limitation, we interleave the channels computed by the first grouped pointwise convolution K. This allows each group of the secondary grouped convolution L to compute data coming from more than one group from the preceding K layer.

Besides, we propose that the output of both grouped convolutions K and L be joined via a summation. Summation has the advantage of not raising the number of output channels compared to concatenation. It also allows the network to learn patterns straight on the first convolution K, skipping the L convolution filters.

Let us note the number of groups in layer i as N_i for grouped convolutions. This number is calculated according to our algorithm, that will be stated below. Each group is fed a subset of C_{i-1}/N_i input channels. The number of filters per group is F_i/N_i. Accordingly, the multiplication of the number of the filters per group and the number of channels per group $(F_i/N_i)\cdot(C_{i-1}/N_i)$ gives the number of parameters per group. The total number of parameters of a grouped convolutional layer can be calculated by multiplying $(F_i/N_i)\cdot(C_{i-1}/N_i)$ by the number of groups N_i, as expressed in Eq. (2):

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

Eq. (2) shows that the number of trainable parameters is inversely proportional to the number of groups. It should be mentioned that we follow the constraints listed below when calculating the number of groups per convolution N_i:

- Each group must have a minimum number of input channels ch. This is the minimum number of input channels that can be operated together by each parallel group across all optimized pointwise convolutions.
- The greatest common divisor of C_{i-1} and F_i determines the number of groups N_i, as long as it respects the previous constraint ($C_{i-1}/N_i \geq ch$).
- The number of groups must be bigger than 1 ($N_i \geq 1$).

For each pointwise convolutional layer in the original architecture, if there is no solution to the above constraints, then the original layer is left as is, without applying the optimization.

An interleaving layer is added when there are two or more output channels (filters) per group ($F_i/N_i \geq 2$). The interleaving is intended to mix channels from the L convolution so any 2 channels from the same group are not placed together. When there is only one output channel per parallel group, the interleaving is not required.

A grouped pointwise convolutional layer L is added after the interleaving layer, when the number of input channels ($C_{i-1} \geq C_i$) is greater than or equal to the number of output channels. The result of both grouped convolutional layers K and L are then added. When the number of input channels is smaller than the number of output channels, there is less chance of input information being lost due to a lack of output channels. In this case, the extra learning capacity supplied by the secondary L grouped convolution is not as important. As a result, we do not utilize an L layer in this case.

To understand how the proposed technique can reduce the number of parameters, let us assume that we have a monolithic pointwise convolution with $C_{i-1} = 1,024$ and $F_i = 512$, which produces $P_i = 524,288$ parameters. By substituting this pointwise convolution with the proposed sub-architecture, employing 16 channels per group, the number of groups will be the number of input channels (1,024) divided by the number of channels per group (16) resulting into $N_i = 64$ groups. In this example, the K grouped convolutional layer will have $1,024\cdot 512/64 = 8,192$ parameters. In the L grouped convolutional layer, the numbers of groups and channels are 64 and 512, respectively. Consequently, the number of parameters will be $512\cdot 512/64 = 4,096$. When these values are added, the total number of parameters for the entire sub-architecture is $8,192 + 4,096 = 12,288$, representing a parameter saving of 97.7%.

The proposed DCNNs variants are deeper than their original counterparts as we add up to 2 convolutions where originally we find one. In contrary to the work of Ting Zhang et. al. [21], besides that we don't have spacial convolutions, we sum the outputs of both grouped convolutions via a skip connection. This added skip connection counter balance the difficulty for gradients to flow in deeper architectures. We also calculate the number of parallel groups in a way that can be applied to any pointwise convolution from any existing architecture. In our proposal, the number of parallel groups vary according to the number of input channels and filters instead of being a constant number.

We tested our optimization by replacing original pointwise convolutions in the EfficientNet-B0, DenseNet-BC L100, Inception V3, MobileNet and MobileNet V3 Large architectures. We name our modified versions as kEffNet-B0, kDenseNet-BC L100, kMobileNet, kMobileNet V3, respectively. EfficientNet-B0, DenseNet-BC L100, and MobileNets were selected for their parameter efficiency, so we can test our ideas on already efficient architectures. In the original architectures, when the last convolutional layer has an 1x1xC activation map as input shape, it is left unmodified as it behaves as a dense layer for the final classification. For the kEffNet-BO, we tested an additional modification that skips the first 4 convolutional strides, which allows input images with 32x32 pixels instead of the original resolution of 224x224 pixels.

We performed our experiments with various hardware configurations with NVIDIA graphics cards. Regarding software, we worked with K-CAI/Keras/Tensorflow 2.* [14, 2, 1] and RMSProp optimizer. All of our experiments have a cyclical learning rate of 25 epochs and data augmentation. For the CIFAR-10 and CIFAR-100 datasets, we used 50 epochs. For the Oxford-IIIT Pet and the Cropped PlantDoc datasets, we trained for 150 and 75 epochs, respectively. To compensate the limited number of images in these datasets, we trained the DCNNs for more than 50 epochs. It is worth noting that the number of epochs is a multiple of our learning rate cycle, which is 25. In this study, we did not employ transfer learning as our main objective is to assess the learning capacity of parameter-efficient DCNNs models. [1]

4 Experimental Results

In this section, we analyze test classification accuracy and class activation maps.

4.1 Analyzing Classification Accuracy

Table 1 compares test accuracies, number of trainable parameters, number of floating-point computations and a percentage of the original number of trainable parameters and computations with the CIFAR-10 experiments. Our variant names always start with a character k. We add the minimum number of input channels per group to the end of the name of our implementations. As an example, kEffNet-B0 32ch has a minimum of 32 input channels per group. Regarding test accuracy, at least one of our variants for EfficientNet, Inception V3 and MobileNet V3 Large achieve higher accuracy than their respective baseline models. For the MobileNet, we have a close result to its baseline. Our DenseNet variants underperform the original DenseNet-BC L100. Our variant with smallest number of parameters that outperforms its baseline is MobileNet V3 Large 32ch. It has a significant reduction of 83% of the trainable parameters and requires 54% of the computations of the original model. Our second smallest architecture that outperform its baseline is the kEffNet-B0 32ch with 32x32 pixels input resolution. It has 26% of the original parameters and requires 35% of the original computations.

In Table 1, overall, our variant that achieves highest classification accuracy is kEffNet-B0 32ch with 224x224 pixels input image resolution. It achieves slightly higher classification accuracy than its baseline, which empirically proves that our reduction algorithm is working as well as its baseline with a fraction of the original resources, i.e., 26% of the trainable parameters and 45% of the computations. The proportion of computations/trainable parameters differ across layers. In general, convolutional layers have more floating point computations per parameter than dense layers. Also, in convolutional layers, the number of computations is proportional to the input resolution. This is why a saving in parameters doesn't exactly result in a proportional saving in computations.

[1] Our source code is publicly available at https://github.com/joaopauloschuler/kEffNetV1/.

The two best performing variants kEffNet-B0 and kMobileNet V3 in Table 1 were retested with CIFAR-100, Cropped PlantDoc and Oxford-IIIT Pet datasets as per Tables 2, 3 and 4 respectively.

In our Cropped PlantDoc experimentation shown in the Table 3, we obtained the highest accuracy with kEffNet-B0 32ch (65.74%) followed by kMobileNet V3 Large 32ch (65.34%). In this table, all of our variants achieved higher test accuracies than their baselines. Our kMobileNet V3 Large 16ch has only 10% of the original trainable parameters, 37% of the original computations and achieves higher accuracy by a margin of 15%. Our kEffNet-B0 16ch has 16% and 33% of the original trainable parameters and computations respectively. It achieves a higher accuracy than its baseline by a margin of 0.5%.

In our Oxford-IIIT Pet dataset experimentation shown in Table 4, we obtained highest accuracy with kEffNet-B0 16ch (64.56%) followed by kMobileNet V3 Large 32ch (63.09%). We speculate that this result is consequence of overfitting due to a small training dataset. In all other tested datasets, the 32ch variant achieves higher accuracy than its 16ch related variant.

4.2 Class Activation Maps

Figure 2 shows CAMs made with the Oxford-IIIT Pet dataset. Our kEffNet-B0 focuses on face, ears and the top of the head. In turn, the EfficientNet-B0 baseline has its focus more frequently in the background.

Figure 3 shows the CAMs of the baseline and our kEffNet-B0. In these cat images, both models have some focus on areas of the background although kEffNet still does a better job at focusing on the cat.

5 Discussion

Indeed, the best performing optimized architectures are based on EfficientNet, MobileNet and MobileNet V3. These 3 architectures share a characteristic in common that explain why our optimization fits well: in the original architectures, the main convolutions are parameter efficient depthwise convolutions and parameter inefficient pointwise convolutions. Given that our replacement is designed specifically for reducing excess of connections in pointwise convolutions, we apply our optimization at the precise weak point from the original architectures.

The variants derived from DenseNet-BC L100 underperformed the original model. In DenseNet-BC, most pointwise convolutions are followed by standard convolutions that intermix all input channels. The added complexity from our architecture to intermix channels becomes redundant as this is done via standard convolutions in the original architecture. This extra redundancy adds complexity and parameters, which becomes a drawback more than an advantage for the training process.

Table 1: CIFAR-10 testing results after 50 epochs. % columns indicate parameters and computation percentages to their original models.

architecture	input size	params	%	computations	%	test acc.
EfficientNet-B0	224x244	4.02M		389.9M		93.52%
kEffNet-B0 16ch	224x224	**0.64M**	**16%**	129.0M	33%	92.24%
kEffNet-B0 32ch	224x224	1.06M	26%	174.5M	45%	**93.75%**
kEffNet-B0 16ch	32x32	**0.64M**	**16%**	**84.8M**	**22%**	92.46%
kEffNet-B0 32ch	32x32	1.06M	26%	138.4M	35%	93.61%
DenseNet-BC L100	32x32	0.77M		288.0M		**92.38%**
kDenseNet-BC L100 12ch	32x32	**0.35M**	**45%**	**138.2M**	**48%**	90.83%
kDenseNet-BC L100 24ch	32x32	0.38M	50%	159.6M	55%	90.63%
Inception V3	224x224	21.79M		2.8B		88.29%
kInception V3 16ch	224x224	**14.88M**	**68%**	**2.3B**	**81%**	91.10%
kInception V3 32ch	224x224	15.01M	69%	2.3B	81%	**91.22%**
MobileNet	224x224	3.22M		567.8M		**93.15%**
kMobileNet 16ch	224x224	**0.24M**	**8%**	**92.0M**	**16%**	89.81%
kMobileNet 32ch	224x224	0.40M	13%	153.8M	27%	91.27%
kMobileNet 64ch	224x224	0.72M	22%	251.8M	44%	92.08%
kMobileNet 128ch	224x224	1.32M	41%	201.4M	35%	93.02%
MobileNet V3 Large	224x224	4.21M		217.5M		92.80%
kMobileNet V3 Large 16ch	224x224	**0.40M**	**10%**	**81M**	**37%**	92.74%
kMobileNet V3 Large 32ch	224x224	0.71M	17%	117.3M	54%	**93.26%**

Table 2: CIFAR-100 results after 50 epochs. % columns indicate parameters and computations percentages to their original models.

architecture	input size	params	%	computations	%	test acc.
EfficientNet-B0	224x244	4.14M		390.1M		**74.23%**
kEffNet-B0 16ch	224x224	**0.75M**	**18%**	**129.1M**	**33%**	71.92%
kEffNet-B0 32ch	224x224	1.17M	28%	174.6M	45%	73.93%
MobileNet V3 Large	224x224	4.33M		217.6M		70.73%
kMobileNet V3 Large 16ch	224x224	**0.52M**	**12%**	**81.1M**	**37%**	71.36%
kMobileNet V3 Large 32ch	224x224	0.83M	19%	117.4M	54%	**73.24%**

Table 3: Cropped PlantDoc testing results after 75 epochs. % columns indicate parameters and computations percentages to their original models.

architecture	input size	params	%	computations	%	test acc.
EfficientNet-B0	224x244	4.04M		390.0M		63.50%
kEffNet-B0 16ch	224x224	**0.66M**	**16%**	**129.0M**	**33%**	64.04%
kEffNet-B0 32ch	224x224	1.08M	27%	174.5M	45%	**65.74%**
MobileNet V3 Large	224x224	4.24M		217.5M		46.45%
kMobileNet V3 Large 16ch	224x224	**0.43**	**10%**	**81.0M**	**37%**	61.65%
kMobileNet V3 Large 32ch	224x224	0.74M	17%	117.3M	54%	**65.34%**

Table 4: Oxford-IIIT Pet testing results after 150 epochs. % columns indicate parameters and computations percentages to their original models.

architecture	input size	params	%	computations	%	test acc.
EfficientNet-B0	224x244	4.11M		390.0M		62.05%
kEffNet-B0 16ch	224x224	**0.67M**	**17%**	**129.0M**	**33%**	**64.56%**
kEffNet-B0 32ch	224x224	1.09M	27%	174.53M	45%	62.92%
MobileNet V3 Large	224x224	4.24M		217.5M		52.21%
kMobileNet V3 Large 16ch	224x224	**0.36M**	**10%**	**81.0M**	**37%**	60.39%
kMobileNet V3 Large 32ch	224x224	0.74M	18%	117.3M	54%	**63.09%**

The proposed pointwise replacement does not directly explain why we obtained better classification accuracy with Cropped-PlantDoc or with the Oxford-IIIT Pet datasets. As we have less trainable parameters, our modified kEffNets are likely less prone to overfitting. In the case of CIFAR-10, the baseline and our modified kEffNet models obtained approximately the same test classification. Our optimization tech-

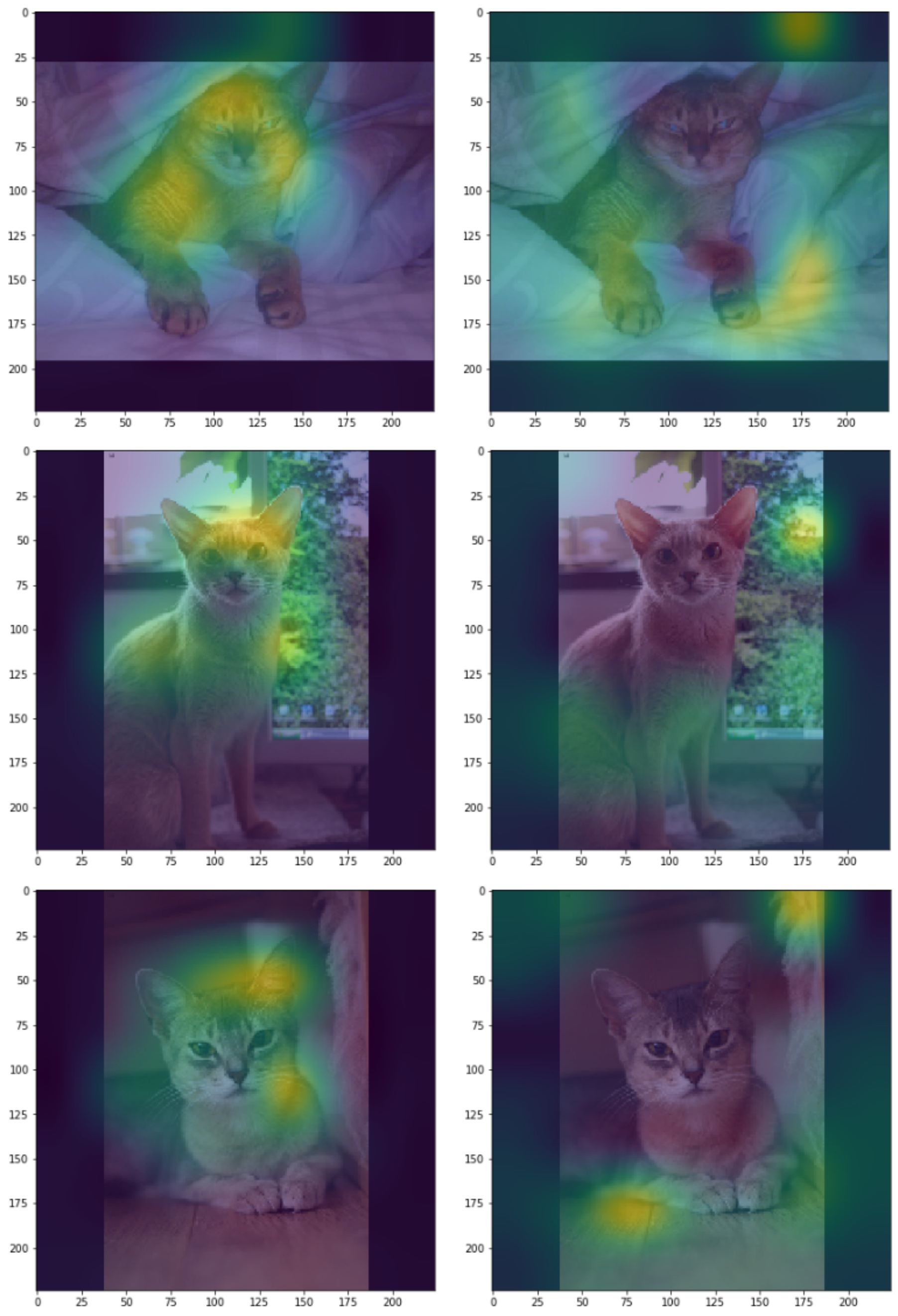

Figure 2: CAMs for images taken from the Oxford-IIIT Pet dataset. (left) CAMs produced by kEffNet-B0 with the proposed pointwise optimization technique. (right) CAMs produced by the EfficientNet-B0 baseline.

nique effectively saves unnecessary connections along the original pointwise convolutions. In contrast, with CIFAR-10, both models (baseline and reduced version) are not excessively degraded by overfitting.

Generated CAMs demonstrated that the baseline tends to focus attention on the background of the images, which may not necessarily be a consequence of overfitting. It may be possible that background features appear more frequently in some image classes than others. The extra parameters of the baselines might be used for these background features.

6 Conclusion

This work presented a parameter and computationally efficient replacement for pointwise convolutions. Specifically, we proposed substituting pointwise convolutions with a sub-architecture comprising two grouped convolutions (K and L) with interleaving and summation layers. As an example, the pointwise convolution optimization of EfficientNet-B0, called kEffNet-B0 32ch, saves 74% of the trainable parameters and 55% of the floating-point computations compared to its baseline. On the CIFAR-10 dataset, our optimized architecture achieves a slightly higher classification ac-

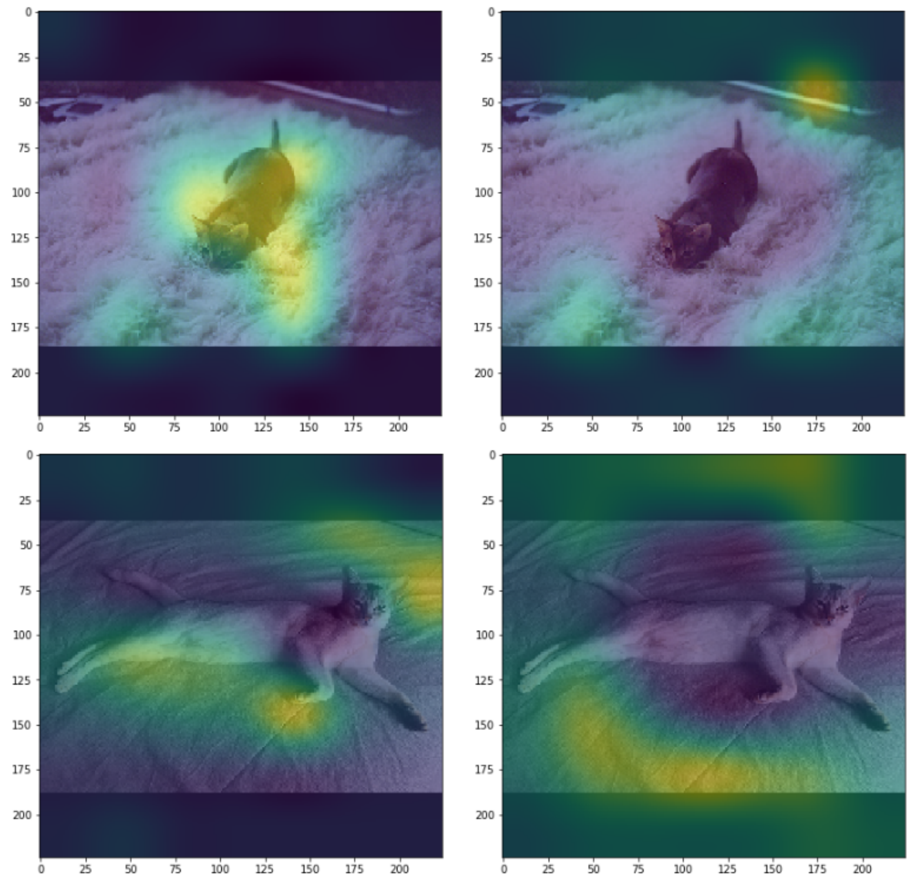

Figure 3: CAMs showing image samples which architectures do not focus on the cat. (left) CAMs produced by kEffNet-B0 with the proposed pointwise optimization technique. (right) CAMs produced by the EfficientNet-B0 baseline.

curacy than the baseline, when trained from scratch. At the light of this and other results shown above, we conclude that the number of connections (parameters) in pointwise convolutions can be significantly reduced without sacrificing any of the original learning capacity. Therefore, we can deduce that most of the original connections in pointwise filters are redundant.

In other specific experiments, we obtained higher accuracy with the Cropped-PlantDoc (+2.24%) and Oxford-IIIT Pet datasets (+1.04%) compared to our baselines. On the CIFAR-100 dataset, our kEffNet-B0 32ch achieved slightly lower classification accuracy (-0.3%) with significantly less parameters (-72%) and floating-point computations (-55%). This results indicate that our optimization works better on architectures with a mix of depthwise and pointwise convolutions such as MobileNet, MobileNet V3 Large, and EfficientNet architectures.

Our experiments confirmed our working hypothesis: to achieve a reasonable degree of pattern recognition, not all input channels need to be connected in every pointwise filter. Parallel groups of pointwise filters can gather subsets of features for proper and efficient image classification.

References

[1] Abadi, M., et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Chollet, F., et al. Keras. https://keras.io, 2015.

[3] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[4] Howard, A., et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 1314–1324.

[5] Howard, A. G., et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* (2017).

[6] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 4700–4708.

[7] Ioannou, Y., Robertson, D., Cipolla, R., and Criminisi, A. Deep roots: Improving cnn ef-

ficiency with hierarchical filter groups. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 1231–1240.

[8] KRIZHEVSKY, A. Learning multiple layers of features from tiny images. Tech. rep., University of Toronto, 2009.

[9] KRIZHEVSKY, A., SUTSKEVER, I., AND HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems 25* (2012).

[10] LIN, M., CHEN, Q., AND YAN, S. Network in network, 2014.

[11] PARKHI, O. M., VEDALDI, A., ZISSERMAN, A., AND JAWAHAR, C. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition* (2012), IEEE, pp. 3498–3505.

[12] RUSSAKOVSKY, O., ET AL. Imagenet large scale visual recognition challenge. *International journal of computer vision 115*, 3 (2015), 211–252.

[13] SCHULER, J., ROMANÍ, S., ABDEL-NASSER, M., RASHWAN, H., AND PUIG, D. *Grouped Pointwise Convolutions Significantly Reduces Parameters in EfficientNet.* 10 2021, pp. 383–391.

[14] SCHULER, J. P. S. K-cai neural api, 2021. https://doi.org/10.5281/zenodo.5810092.

[15] SIMONYAN, K., AND ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[16] SINGH, D., JAIN, N., JAIN, P., KAYAL, P., KUMAWAT, S., AND BATRA, N. Plantdoc: A dataset for visual plant disease detection. In *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD* (2020), pp. 249–253.

[17] SZEGEDY, C., ET AL. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 1–9.

[18] TAN, M., AND LE, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning* (2019), PMLR, pp. 6105–6114.

[19] XIE, S., GIRSHICK, R., DOLLÁR, P., TU, Z., AND HE, K. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 1492–1500.

[20] ZEILER, M. D., AND FERGUS, R. Visualizing and understanding convolutional networks. In *European conference on computer vision* (2014), Springer, pp. 818–833.

[21] ZHANG, T., QI, G., XIAO, B., AND WANG, J. Interleaved group convolutions for deep neural networks. *arXiv preprint arXiv:1707.02725* (2017).

[22] ZHOU, B., KHOSLA, A., LAPEDRIZA, A., OLIVA, A., AND TORRALBA, A. Learning deep features for discriminative localization.