EFFECTIVE APPROACHES FOR IMPROVING THE EFFICIENCY OF DEEP CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE CLASSIFICATION

Joao Paulo Schwarz Schuler

UNIVERSITAT ROVIRA i VIRGILI

Effective Approaches for Improving the Efficiency of Deep Convolutional Neural Networks for Image Classification

JOAO PAULO SCHWARZ SCHULER

DOCTORAL THESIS
2022

UNIVERSITAT ROVIRA I VIRGILI
EFFECTIVE APPROACHES FOR IMPROVING THE EFFICIENCY OF DEEP CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE CLASSIFICATION
Joao Paulo Schwarz Schuler

Effective Approaches for Improving the Efficiency of Deep Convolutional Neural Networks for Image Classification

DOCTORAL THESIS

Author:
Joao Paulo Schwarz Schuler

Advisors:
Dr. Santiago Romaní Also
Dr. Mohamed AbdelNasser Mohamed Mahmoud
Dr. Domènec Savi Puig Valls

Departament d'Enginyeria Informàtica i Matemàtiques

UNIVERSITAT ROVIRA I VIRGILI
Tarragona
2022

UNIVERSITAT ROVIRA I VIRGILI
EFFECTIVE APPROACHES FOR IMPROVING THE EFFICIENCY OF DEEP CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE CLASSIFICATION
Joao Paulo Schwarz Schuler

UNIVERSITAT
ROVIRA I VIRGILI
Departament d'Enginyeria Informàtica
i Matemàtiques

Av. Paisos Catalans, 27
43007 Tarragona
Tel. +34 977 55 95 95
Fax. +34 977 55 95 97

We STATE that the present study, entitled "Effective Approaches for Improving the Efficiency of Deep Convolutional Neural Networks for Image Classification", presented by Joao Paulo Schwarz Schuler, for the award of the degree of Doctor, has been carried out under our supervision at the Departament d'Enginyeria Informàtica i Matemàtiques.

Tarragona, 26^{th} September 2022.

Doctoral Thesis Supervisors,
Dr. Santiago Romaní Also

Dr. Mohamed AbdelNasser Mohamed Mahmoud

Dr. Domènec Savi Puig Valls

UNIVERSITAT ROVIRA I VIRGILI
EFFECTIVE APPROACHES FOR IMPROVING THE EFFICIENCY OF DEEP CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE CLASSIFICATION
Joao Paulo Schwarz Schuler

To the Spanish people for allowing me to study in Spain. To Santiago for being the most dedicated professor that I have ever had. To Alejandra for convincing me to study at URV.

UNIVERSITAT ROVIRA I VIRGILI
EFFECTIVE APPROACHES FOR IMPROVING THE EFFICIENCY OF DEEP CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE CLASSIFICATION
Joao Paulo Schwarz Schuler

Abstract

Recent architectures in Deep Convolutional Neural Networks (DCNNs) have a very high number of trainable parameters and, consequently, require plenty of hardware and time to run. It's also commonly found in the literature that most parameters in a DCNN are redundant. This thesis presents two methods for reducing the number of parameters and floating-point computations in existing DCNN architectures applied for image classification. The first method reduces parameters in the first layers of a neural network, while the second method reduces parameters in deeper layers.

The first method is a modification of the first layers of a DCNN that splits the channels of an image encoded with CIE Lab color space in two separate branches, one for the achromatic channel and another for the remaining chromatic channels. We modified an Inception V3 architecture to include one branch specific for achromatic data (L channel) and another branch specific for chromatic data (AB channels). This modification takes advantage of the decoupling of chromatic and achromatic information. Besides, splitting branches reduces the number of trainable parameters and computation load by up to 50% of the original figures in the modified layers. We achieved a state-of-the-art classification accuracy of 99.48% on the PlantVillage dataset. This thesis also shows that this two-branch method improves image classification reliability when the input images contain noise.

Besides the first layers in a DCNN, in deeper layers of some recent DCNN architectures, more than 80% of the parameters come from standard pointwise convolutions. The parameter count in pointwise convolutions quickly grows due to the multiplication of the filters and input channels from the preceding layer. The second optimization method introduced in this thesis is making pointwise convolutions parameter-efficient via parallel branching to handle this growth. 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. We tested our optimization on an EfficientNet-B0 as a baseline architecture and made classification tests on the CIFAR-10, Colorectal Cancer

Histology, and Malaria datasets. For each dataset, our optimization saves 76%, 89%, and 91% of the number of trainable parameters of EfficientNet-B0, while keeping its test classification accuracy.

Keywords: EfficientNet, Deep Learning, Computer Vision, Image Classification, Convolutional Neural Network, CNN, DCNN, Grouped Convolution, Pointwise Convolution, Neural Network Optimization, Parameter Reduction, Parallel Branches, Channel Interleaving.

Acknowledgements

The Spanish Government partly supported this research through Project PID2019-105789RB-I00, allowing the paper *An Enhanced Scheme for Reducing the Complexity of Pointwise Convolutions in CNNs for Image Classification Based on Interleaved Grouped Filters without Divisibility Constraints* (Schwarz Schuler et al. (2022a)) to be published in open access format.

I thank Dr. Santiago Romaní for his extreme dedication towards work, Dr. Domenèc Puig for his guidance, Dr. Mohamed Abdel-Nasser for his always positive and motivational mindset and Hatem Rashwan for his problem-solving skills.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

This introductory chapter comments on some major scientific achievements in artificial intelligence relevant to this thesis. The datasets used in this thesis are detailed. This chapter also presents this thesis's scientific dissemination via already published papers. Finally, the thesis organization is described.

1.1 Deep learning

Fukushima (1980) devised a layered artificial neural network inspired by the visual cortex structure for image classification. Such a network showed that the first layer contains neurons detecting simpler patterns with a small receptive field. Deeper layers detect more complex patterns with wider receptive fields by composing patterns from previous layers.

LeCun et al. (1989a) devised the first Convolutional Neural Network (CNN), which mimicked the organization of neural cells in the visual cortex as convolutional filters. This new type of neural network could accurately recognize 10 digits in hand-written text.

Krizhevsky et al. (2012) achieved a breakthrough in the ImageNet Large Scale Visual Recognition Challenge with their AlexNet architecture. Since then, many other CNN architectures have been introduced: ZFNet Zeiler (2014), VGG Simonyan and Zisserman (2015), GoogLeNet Szegedy et al. (2015), ResNet He et al. (2016a), DenseNet Huang et al. (2017) and others. Since the number of layers has increased from 5 to more than 200, those models are called "deep learning".

1.2 CNNs and color spaces

Most existing CNNs are trained with the basic Red-Green-Blue (RGB) color values of input pixels. Despite this being the obvious choice taking into account that digital images are usually encoded in RGB, it is curious that very few researchers have attempted to train their networks on images encoded with other color spaces such as Hue-Saturation-Lightness (HSL) or CIE-LAB, the definition of which are vastly known and long-standing in the fields of color perception Robertson (1992) and colorimetry Wyszecki et al. (1982).

1.3 Pointwise convolutions

Pointwise convolutions have 1x1 kernels with one trainable parameter per input channel. These kernels do not consider neighboring positions such for example, 3x3 filters. Each filter in typical pointwise convolutions has one trainable parameter per input channel. It is important to note that more than 80% of the parameters in the most recent CNN architectures are found in pointwise convolutions.

1.4 Pruning methods

Most parameters in DCNNs are redundant Denil et al. (2013); Cheng et al. (2015); Yang et al. (2019); Kahatapitiya and Rodrigo (2021); Liebenwein et al. (2021).

Existing pruning methods remove connections and neurons found irrelevant by different techniques. After training the original network with the full set of connections, the removal is done LeCun et al. (1989b); Reed (1993); Zhuang et al. (2018); Han et al. (2016); Baykal et al. (2019); Liebenwein et al. (2020).

1.5 Thesis objectives

The main objective of this thesis is to reduce the number of parameters and floating-point operations in DCNNs while maintaining or improving classification accuracy.

Our working hypothesis is that such reduction can be achieved effectively through filter grouping inside some specific layers. This means that all filters (neurons) in the target layer are split into parallel branches, each dealing with a subset of the input channels. This prevents many neural connections from being formed, which is not convenient. However, the reduced set of connections in each branch can specialize in recognizing specific features, which leads to a functional DCNN that needs significantly fewer resources (fewer parameters and fewer floating point calculations).

In this thesis, we exploit filter grouping in the first filters of the DCNNs, based on the color separation into achromatic and chromatic channels and the deeper pointwise convolutional layers.

The approaches proposed in this thesis differ from existing pruning as we reduce the number of connections before the training starts, while pruning does after training. Therefore, our methods can save computing resources during training time.

1.6 Scientific dissemination

The work presented in this thesis has been disseminated via 5 open access papers, three journal papers, and two conference papers.

1.6.1 Journal articles

One paper was published in the Entropy Journal, and another two papers were published in Mendel Journal.

The following paper was published in the Entropy journal:

Joao Paulo Schwarz Schuler, Santiago Romani, Domenec Puig, Hatem Rashwan, Mohamed Abdel-Nasser ***An Enhanced Scheme for Reducing the Complexity of Pointwise Convolutions in CNNs for Image Classification Based on Interleaved Grouped Filters without Divisibility Constraints***, Entropy. (Schwarz Schuler et al. (2022a)).

According to ISI-JCR, the entropy journal has an impact factor of 2.738 (Q2, Physics, multidisciplinary).

Two papers were published in the Mendel journal:

1. Joao Paulo Schwarz Schuler, Santiago Romani, Mohamed Abdel-Nasser, Hatem Rashwan, Domenec Puig, ***Grouped Pointwise Convolutions Reduce Parameters in Convolutional Neural Networks***, Mendel. (Schwarz Schuler et al. (2022c))
2. Joao Paulo Schwarz Schuler, Santiago Romani, Mohamed Abdel-Nasser, Hatem Rashwan, Domenec Puig, ***Color-Aware Two-Branch DCNN for Efficient Plant Disease Classification***, Mendel. (Schwarz Schuler et al. (2022b)).

According to SJR Scimago, the Mendel journal has an SJR of 0.2 (Q4, Artificial Intelligence).

1.6.2 Conference Papers

Two papers were presented at 23^{rd} International Conference of the Catalan Association for Artificial Intelligence (CCIA2021):

1. Joao Paulo Schwarz Schuler, Santiago Romani, Mohamed Abdel-Nasser, Hatem Rashwan, Domenec Puig, ***Reliable Deep Learning Plant Leaf Disease Classification Based on Light-Chroma Separated Branches***, Artificial Intelligence Research and Development, pp. 375-382, IOS Press, 2021. (Schuler et al. (2021b))
2. Joao Paulo Schwarz Schuler, Santiago Romani, Mohamed Abdel-Nasser, Hatem Rashwan, Domenec Puig, ***Grouped Pointwise Convolutions Significantly Reduces Parameters in EfficientNet***, Artificial Intelligence Research and Development, pp. 383-391, IOS Press, 2021. (Schuler et al. (2021a))

1.7 Thesis organization

The thesis is outlined as follows. In chapter 2, we analyze a CNN classifying the CIFAR-10 dataset encoded in the RGB, HSV, HSL, and CIE-LAB color spaces. We show that CNNs can learn filters dedicated to chromatic and achromatic patterns. We devised a modification of the first layer that splits the channels of an image encoded with HSL, LAB, or similar color space into two separate paths, one for the achromatic channel and another for the remaining chromatic channels. This change reduces DenseNet-BC L40 forward pass floating point computations by 33% while maintaining the baseline classification accuracy. This chapter gives the basis for the following two chapters.

In chapter 3, inspired on chapter 2, we modified an Inception V3 to process CIE Lab encoded images. We created one branch specific for achromatic data (L channel) and another for chromatic data (AB channels). In the modified layers, we reduced the number of trainable parameters and computation load by up to 50%. We achieved a state-of-the-art classification accuracy of 99.48% on the PlantVillage

dataset and 76.91% on the Cropped-PlantDoc dataset. The results of this chapter are published in Schwarz Schuler et al. (2022b).

In chapter 4, we show that the two-branch Inception V3 introduced in chapter 3 provides better classification reliability when perturbing the images with noise (salt and pepper, blurring, motion blurring, and occlusions). We hypothesize that the filters in the AB branch provide noise resistance due to their relatively low frequency in the image-space domain. The results of this chapter are published in Schuler et al. (2021b).

In chapters 2, 3 and 4, we show an optimization done along the first layers of a CNN. In contrast, chapter 5 shows an optimization done in deeper layers of a CNN. In some well-known CNN architectures, most parameters are found in pointwise convolutions. The number of parameters in pointwise convolutions quickly grows due to the multiplication of the filters and input channels from the preceding layer. To mitigate this growth by grouping filters, chapter 5 proposes a new technique that makes pointwise convolutions parameter-efficient. We tested this optimization with EfficientNet, DenseNet-BC L100, MobileNet, and MobileNet V3 Large architectures. The results of this chapter are published in Schwarz Schuler et al. (2022b).

Chapter 6 refines the algorithm shown in chapter 5 so groups of filters can cope with non-divisible numbers of input channels and filters. This refined method further reduces the number of floating-point computations (11%) and trainable parameters (10%) achieved in the previous chapter. We made classification tests on the CIFAR-10, Colorectal Cancer Histology, and Malaria datasets. For each dataset, our optimization saves 76%, 89%, and 91% of the number of trainable parameters of EfficientNet-B0, while keeping its test classification accuracy. Most results of this chapter are published in Schwarz Schuler et al. (2022a).

Chapter 7 presents the thesis's conclusion and future research lines.

References are presented at the end of the thesis.

CHAPTER 2

L+AB branches - Introductory example

In this chapter, we have experimented with minimal CNN architectures to analyze their behavior in front of different color encoding of input images. Specifically, we have tested the CIFAR-10 classification task on images encoded with Grayscale and RGB, HSV, HSL, and CIE-LAB color spaces. Our aim is not to find an optimal classification network but to acquire insights into how CNNs deal with color-related information. As one of the main contributions, we show that CNNs can learn some filters dedicated to achromatic patterns independently from other filters dedicated to chromatic patterns, achieving similar results in all color spaces. We have also devised a modification of the first layer that splits the channels of an image encoded with HSL, LAB, or alike color space in two separate paths, one for the achromatic channel and another for the remaining chromatic channels, which reduces DenseNet-BC L40 forward pass floating-point computation in 33% while maintaining the baseline classification accuracy.

2.1 Introduction

The work of Zeiler (2014) is interesting to this thesis as they performed an in-depth study of the content of the convolutional filters of all layers after training the AlexNet and their ZFNet for the ImageNet dataset. Visualizing the obtained weights gave clues for understanding which features are learned by the convolutional filters, from small details in the first layer (yet pointed out in Krizhevsky et al. (2012)) to object parts in the top layers.

The rationale behind trying other color spaces than RGB is based on the evidence that the human color vision transforms the initial neural signals from cones and rods into an opponent color model Hering (1920), where several layers of neurons convert the Short (CS), Medium (CM) and Large wavelength (CL) neural signals, loosely related to blue, green and red hues, into other neural signals, symbolically stated in equation 2.1:

$$\begin{aligned} GR &= \alpha_1 \cdot CM - \alpha_2 \cdot CL \\ BY &= \beta_1 \cdot CS - \beta_2 \cdot CM - \beta_3 \cdot CL \\ WB &= \gamma_1 \cdot CS + \gamma_2 \cdot CM + \gamma_3 \cdot CL \end{aligned} \tag{2.1}$$

Equation 2.1 is an empiric model of the reinforcement (addition) and inhibition (subtraction) of neural signals, which results in two chromatic signals contrasting green against red features (GR) and blue against yellow features (BY) of a color stimulus. An achromatic channel (WB) combines all primary signals to obtain brightness information (white and black). From the point of view of human color perception Robertson (1992), these opponent signals are further processed and converted into perceptual color components named Hue, Saturation, and Lightness. Several computational models convert RGB into HSL-related components, for example, Smith's HSI Smith (1978) and Yagi's HSV Nakatani (1942).

From the point of view of colorimetry Wyszecki et al. (1982), CIE Lab is one of the most common color coordinates where color differences appreciated by humans are uniform throughout the color space. From a mathematical point of view, RGB components are highly correlated. An increase in light intensity affects all three RGB

channels proportionally. Therefore, we can search for a linear transformation that maps RGB values into another 3D space defined by uncorrelated eigenvectors. For example, Otha et al. Ohta et al. (1980) defined the I1-I2-I3 color space utilizing a Karhunen-Loewe analysis of some RGB color samples extracted from eight natural images. The obtained transformation is expressed in equation 2.2, where the first uncorrelated direction, I1, is defined by the diagonal (light) of the RGB space:

$$\begin{aligned} I1 &= (R+G+B)/3 \\ I2 &= (R-B)/2 \\ I3 &= (2G-R-B)/4 \end{aligned} \tag{2.2}$$

In general, we can observe that most color representations based on human perception or mathematical significance of color information provide a three-component color space, where one of the components is achromatic (WB, L, V, I1). The other two carry the chromaticity of the color (GR/BY, H/S, A/B, I2/I3). Therefore, it seems evident that any CNN design may be improved by transforming the RGB channels of an input image into achromatic/chromatic channels. Hence the filters may specialize in achromatic/chromatic features of the images, similar to the human visual system.

The hypothesis behind this proposal is that CNNs may work more efficiently in learning objects in the input images when dealing with these two types of information separately since they respond to different types of visual phenomena: variations in the achromatic channel usually stand for light intensity changes due to shadows, shading or texture, while variations in the chromatic channels usually stand for the intrinsic color of the object surfaces (materials), which may indicate the limits of object parts or the limit of one object in front of another object or background Shafer (1992); Romaní (2006); Gevers et al. (2012).

Few papers have addressed the design and evaluation of CNNs learning color components like HSL or LAB Sutherland (1982); Napoletano (2017); Bianco et al. (2017); Mody et al. (2017). Besides, only a tiny subset evaluates the performance under several color spaces. For example, Cheng et al. Cheng and Guo (2017) modeled

a CNN for rock granularity classification using either HSV, YCbCr, or RGB pixel values. They reported very high accuracies (above 98%) in all three color spaces, with a slight improvement of 1.9% of HSV over RGB. Still, it is not conclusive proof for our hypothesis since we suspect that texture features are more relevant than color in the given image dataset. Hence the referred CNN may have worked similarly with gray-level images. Gowda and Yuan (2019) experimented with CIFAR-10 image classification with RGB, HSV, YUV, YIQ, XYZ, YPbPr, YCbCr, HED, and LCH and CIE Lab color spaces and achieved higher classification accuracy with CIE Lab color space.

2.2 Methodology

To check our hypothesis, we will perform image classification experiments on the CIFAR-10 dataset Krizhevsky (2009), which consists of 60k 32x32 RGB labeled images belonging to 10 different classes: airplane, automobile, bird, cat, etc. These images are taken in natural and uncontrolled lighting environments and contain only one prominent instance of the object to which the class refers. The object may be partially occluded or seen from an unusual viewpoint.

Firstly, we aim to explore a small CNN able to obtain a reasonable test accuracy (above 80%) in the CIFAR-10 image classification task to compare its behavior (accuracy variation, patterns in first layer filters, etc.) with various color encodings from the basic RGB to HSL or LAB, as well as the bare gray level value of pixels (colorless images). Secondly, we aim to retest our hypothesis with a DenseNet architecture Huang et al. (2017) as it promotes feature reuse and provides good classification accuracy.

2.2.1 Small architecture

As a baseline, we defined a single-branch CNN architecture small enough to classify the CIFAR-10 dataset with at least 80% test accuracy in less than one hour. This single-branch architecture is described in section 2.2.1.1. In section 2.2.1.2, we

describe a two-branch CNN architecture that separates achromatic from chromatic components. Common settings for both architectures are described in section 2.2.1.3.

2.2.1.1 Baseline single-branch CNN architecture

We have tested several configurations for our baseline CNN: 2, 4, 8, and 12 inner convolutional layers with 2, 4, 8, and 12 inner fully connected layers. The number of filters in each convolutional layer has also been scanned with 16, 32, 64, 96, and 128 filters per layer. We have found that the results are good enough for our purposes with only 2 inner convolutional layers and 2 inner fully connected layers. Therefore, we propose to analyze the outputs of a basic CNN composed of the layers detailed in table 2.1 and shown in figure 2.1.

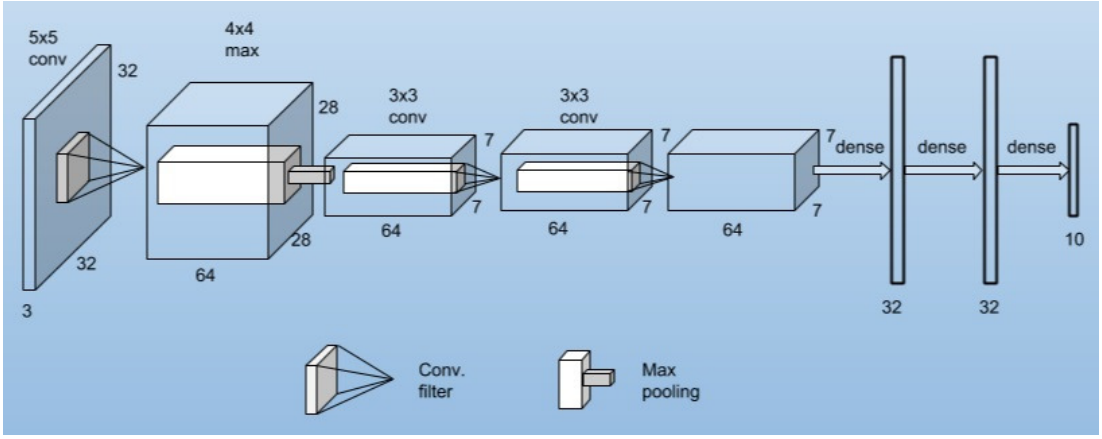

Figure 2.1: Graphical representation of the single-branch baseline CNN architecture

Stage	Input size	Filter size	Stride	Padding	Number of filters	Activation function
1st conv.	32x32x{3,1}	5x5	1x1	0	64	ReLU
Max. pool.	28x28x64	4x4	4x4	0	–	–
2nd conv.	7x7x64	3x3	1x1	1	64	ReLU
3nd conv.	7x7x64	3x3	1x1	1	64	ReLU
1st dense	3136	–	–	–	32	ReLU
2nd dense	32	–	–	–	32	ReLU
3nd dense	32	–	–	–	10	Softmax

Table 2.1: Single-branch baseline CNN detailed architecture layer by layer. The input size is given as height x width x number of channels. Padding is given as the number added to each side. Filter size is given as height x width pixels.

We tested this architecture with RGB, HSV, HSL, and CIE-LAB color

components, as well as with the gray-scale (GRAY) transformation of the input images. GRAY images contain one single channel, while, for color components, images contain three channels. Since the number of channels processed by the first layer filters must adapt to the input, the first convolution input size is set accordingly to 32x32x1 or to 32x32x3 (see table 2.1).

2.2.1.2 Two-branch chromaticity-aware architecture

Inspired by Deep Roots Ioannou et al. (2017), multipath convolutional neural networks Wang (2015) and dual path neural networks Chen et al. (2017), we create parallel branches for better learning of color features, so we split the first layers of our CNN into two parallel branches, one dedicated to achromatic data (L) and the other to chromatic data (AB). We intend to take advantage of separated chromatic and achromatic channels, which are readily available in color spaces such as CIE Lab.

To this aim, we propose to create two separate paths for the first convolutional layer, each one dedicated to each type of pixel information (achromatic/chromatic), to specialize the first layer filters of the CNN to the mentioned aspects of the scene (light variations, object boundaries). We hypothesize that this specialization may lead to better object identification due to a more object-related representation of the image.

The chromatic channels (such as the AB channels in the CIE Lab) should not be further split into separate channels because the chromaticity of the pixels is encoded in both coordinates simultaneously.

Figure 2.2 shows the proposed two-branch architecture, where the top branch processes the single achromatic channel while the bottom branch processes the two chromatic channels. For example, we can convert RGB into CIE-LAB color encoding. Hence the L channel is fed into the top branch, while the AB channels are fed into the bottom branch. In the case of the HSL/HSV color spaces, the L or V channel goes to the top branch, while the HS channels go to the bottom branch.

In this framework, any input image is split into a 32x32x1 achromatic volume

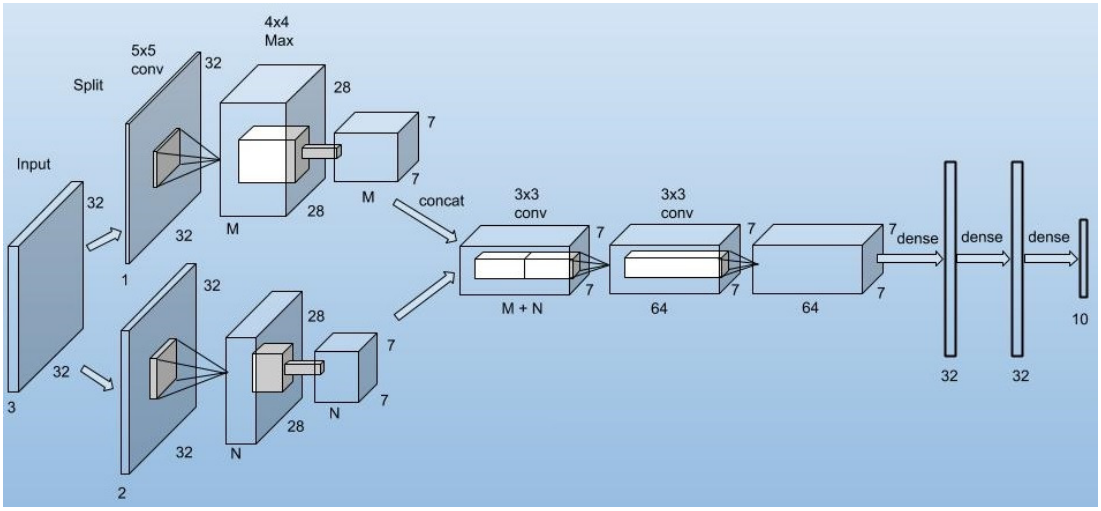

Figure 2.2: Graphical representation of the two-branch CNN architecture.

and a 32x32x2 chromatic volume. Achromatic filters have 5x5x1 weights, while chromatic filters have 5x5x2 weights. We define M achromatic and N chromatic filters, keeping the sum of M plus N equal to 64 for coherence with the baseline model. The activation maps generated by each type of filter are max pooled into 7x7xM and 7x7xN volumes, respectively, which are then concatenated into a single 7x7x64 volume. Experimenting with various M and N values allows us to find an optimal distribution of the CNN's representational power for achromatic and chromatic information.

2.2.1.3 Implementation Details

We trained our CNNs from scratch and stopped the training at 350 epochs, as training and test accuracies/losses used to plateau after 300 epochs. We picked 128 samples per batch and Momentum Gradient Descent with $\gamma = 0.9$ following the AlexNet approach Krizhevsky et al. (2012). The learning rate is initialized to 0.001 in all layers, multiplied by 0.6 every 50 epochs. Learning rate initialization and decay parameters were found via experimentation. We used 40,000 images for training, 10,000 images for validation, and another 10,000 for testing, with a balanced distribution of classes in all subsets, as it is split in the original database.

In all color spaces, values in image channels were scaled into the interval [-2, 2]

using 32 bits of single floating-point values. To combat overfitting, we have used several data augmentation methods on the training subset:

- Images are flopped (mirror-reversed across the vertical axis) with a probability of 50%. This procedure doubles the points of view of the training objects.
- Images are randomly cropped and then resized back to the original size of 32x32 pixels. For each axis, a random number of pixels (from 0 to 8) is cropped. Hence, we may find images with sizes such as 32x24, 27x30, or 26x26. When resizing back to 32x32, images might look stretched on one axis and expanded on the other. The initial position from where the original image is cropped is also random. This procedure provides slight re-scaling of training objects in both axes, fostering a certain degree of scale-invariance for object detection.
- Images are made gray with a probability of 25%. This procedure turns off color information from the training images, which enforces the learning of achromatic features. Other probabilities such as 0% (gray transformation disabled), 12%, and 50% have been tested, but 25% has provided the best results.

2.2.2 DenseNet

We've chosen a small DenseNet-BC L40 with data augmentation for CIFAR-10 classification as a baseline so we could run experiments quickly. Our DenseNet parameters are growth rate k = 12, bottleneck B = 48, compression C = 0.5 and 24 filters in the first convolutional layer.

2.2.2.1 Two-path DenseNet-BC L40

We replaced all convolutions before the first transition block with two branches: the achromatic branch has a first layer with 16 filters followed by convolutions with growth rate k = 6 and bottleneck B = 24. The chromatic branch has 8 filters in the first layer, followed by convolutions with k = 6 and B = 24. Both branches are then concatenated before the first transition block.

2.3 Results

We first collected results from our small architecture as we could run it in less than one hour, allowing us to analyze our hypothesis faster. Our DenseNet implementation takes longer to run; therefore, we analyze it as a second step.

2.3.1 Small single-branch baseline

Figure 2.3 shows the 64 first layer filters obtained by our single branch CNN trained with LAB values (fig. 2.3a and 2.3b) and with the original RGB values (fig. 2.3c).

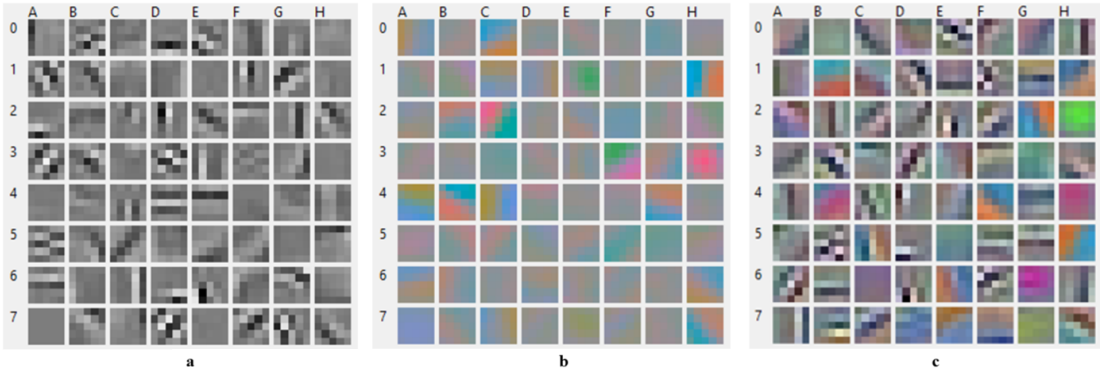

Figure 2.3: Full sets of 64 filters: a) L channel from LAB filters represented in grayscale; b) AB channels from LAB filters. c) Baseline RGB filters.

While in fig. 2.3c each filter is visualized using the obtained 3-channel weights as RGB colors; for LAB filters, we have separated the visual representation of the L weights (fig. 2.3a) from the AB weights (fig. 2.3b), to make it clear the patterns learned by the filters in each subspace (achromatic and chromatic). In fig. 2.3a, the L weights are rendered as gray levels. In fig. 2.3b, the AB weights are concatenated with the maximum value for the L range and then converted back to RGB to enhance its visualization.

In this way, it is clear that some neurons (filters) converge towards color pattern detection, while other neurons converge to gray-level geometric pattern detection, similar to Gabor filters Glorot and Bengio (2010). For example, in fig. 2.3b, filters E1 (green), H3 (red), A7 (bluish), and E7 (yellowish) have converged into one-color filters. In contrast, filters C0, H1, C2, F3, and A4 have converged into two-opponent

color filters, similar to the human eye's opponent color perception. Looking at the same filters in the corresponding cells of fig. 2.3a, the weights in the L channel are generally neutral (uniform mid-level gray) when the weights in the AB channels are active (see E1, H3, A4, and A7 in fig. 2.3a). Conversely, filters in fig. 2.3 clearly shows a strong edge/texture detector in the achromatic channel, like B0, G1, D2, or D7, rendering neutral values in the corresponding cells from fig. 2.3b. In some cases, however, both achromatic and chromatic channels show distinctive patterns, like C2, C4, and H7.

Besides, one can find similar filters in figures 2.3a and 2.3b with respect to figure 2.3c. For example, achromatic filter A3 in fig. 2.3a is equivalent to filter A6 in fig. 2.4c, while chromatic filter C0 in fig. 2.3b is fairly equivalent to filter H1 in fig. 2.3c. Most filters do not have a counterpart between RGB and LAB runs, but they represent functional subsets of possible edge and color patterns in different orientations. For example, filter H1 in fig. 2.3b is the opposite color contrast than filter H5 in fig. 2.3c, hence they can measure the same low-level visual feature but with opposite signs in the neural activation.

Branches	Color space	1st layer filters	1st layer weights	Flops	Val. acc.
1	GRAY	64 (1ch)	1600	2.8M	80.9%
1	RGB	64 (3ch)	4800	8.5M	84.4%
1	HSV	64 (3ch)	4800	8.5M	83.7%
1	HSL	64 (3ch)	4800	8.5M	82.5%
1	LAB	64 (3ch)	4800	8.5M	84.7%
2	LAB	3(1ch)+61(2ch)	3125	5.5M (123k+5.4M)	79.3%
2	LAB	11(1ch)+53(2ch)	2925	5.2M (484k+4.7M)	84.0%
2	LAB	22(1ch)+42(2ch)	2650	4.7M (1.0M+3.7M)	84.1%
2	LAB	32(1ch)+32(2ch)	2400	4.3M (1.4M+2.9M)	84.7%
2	LAB	42(1ch)+22(2ch)	2150	3.9M (1.9M+2.0M)	84.8%
2	LAB	53(1ch)+11(2ch)	1875	3.3M (2.3M+1.0M)	84.1%
2	LAB	61(1ch)+3(2ch)	1675	3.0M (2.7M+267k)	83.1%

Table 2.2: Small Single branch and two-branches configurations.

As stated in table 2.2, the validation accuracy obtained with LAB is 84.7%, which is very similar to the 84.4% obtained with RGB. Experiments with HSV and HSL have reported similar results in filter appearance (chromatic and achromatic patterns) and validation accuracy (83.7% and 82.5%, respectively), although some performance decay is appreciated. We think this decay may be due to the

non-linearity problems of the HSL-like spaces since Hue is circular (highest and lowest values are similar colors) and is also very unstable for low saturation values.

In general, we can say that the baseline architecture has found an optimal set of filters in every color space. Most importantly, the filters in RGB space have naturally specialized into chromatic and achromatic filters, as in LAB, HSV, and HSL spaces. Therefore, the fact that RGB components are correlated with each other is not a problem for CNNs, which can learn achromatic, chromatic, and hybrid patterns.

2.3.2 Small Two-branch L+AB CNN architecture

Figure 2.4 shows two sets of achromatic/chromatic filters obtained by our two-branch architecture (see section 2.2.1.2) trained with LAB values transformed from the original RGB values.

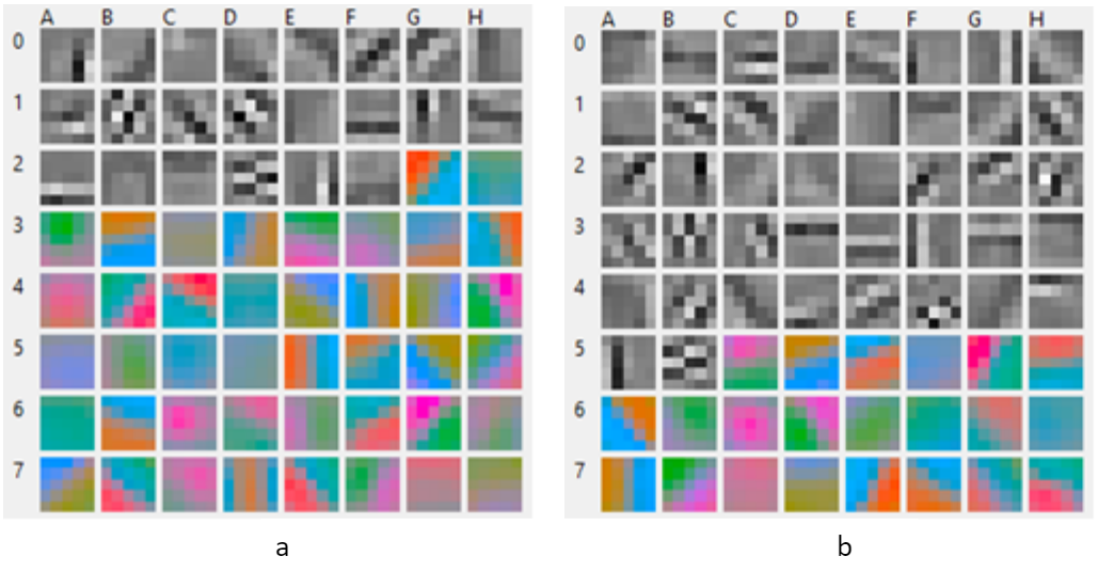

Figure 2.4: Two sets of L+AB filters: a) 22 L + 42 AB; b) 42 L + 22 AB.

In both fig. 2.4a and fig. 2.4b, the visualization of achromatic and chromatic filters have been joined into a single matrix of 64 cells for compactness, although both types of filters do not share the weights of the other type because they are located in different branches. Hence, it is not possible to generate inactive weights in the other subset of channels, as in the case of the one-branch architecture (section 2.3.1). For

this reason, almost all filters in figure 2.4 are active, besides rare exceptions like C3 in Fig. 2.4a.

Figure 2.4 renders patterns similar to the ones shown in figure 2.3 for both achromatic and chromatic features. For example, the achromatic subset presents the typical Gabor-like patterns for edges and texture detection, like G0, G1, and D2 in fig. 2.4a, which are also present in G2, B2, and B5 in fig. 2.4b, and also in G1, D2, and A5 in fig. 2.3a, respectively. We could also find some similar filters in fig. 2.3c (RGB filters), as F6, D4, and D7, although the last one is a hybrid filter, sensitive to the achromatic pattern and a chromatic bluish shade.

In figure 2.4a, we find one-chromatic filters, like A4, A5, and A6, which are similar to C7, E6, and F5 in fig. 2.4b. In the case of two-opponent chromaticity filters, H5 in fig. 2.4a is similar to B7 in fig. 2.4b, and also to F3 in fig. 2.3b.

We have checked different splits of L and AB filters in the first layer of the two-branch architecture to find the best balance for achromatic/chromatic filters at the CIFAR-10 classification task. The visual comparison of subsets of filters in fig. 2.4a and fig. 2.4b is not conclusive, although it seems that 42 chromatic filters may be too many because some patterns in fig. 2.4a are similar or color-inverted, like D4:H2, B7:E7, and E4:G5 (inverted colors). On the other hand, we can also detect some degree of repetition in the achromatic filters of fig. 2.4b, like E3:G3.

According to the validation accuracies reported in table 2.2, the 42 L + 22 AB configuration is slightly superior (+0.7%) to the 22 L + 42 AB configuration. Increasing the proportion of L filters does not improve the accuracy, nevertheless. An equal number of filters for L and AB (32) provides the second best accuracy (84.7%), but it needs 250 more weights than the 42 L + 22 AB configuration. Hence the latter is optimal.

Besides, the best accuracy obtained with the two-branch architecture is similar to that obtained with the one-branch architecture (64 LAB). Still, the former only uses 45% of the number of weights needed in the latter, thus saving 55% of them. This weight reduction does not decrease the performance since we are getting rid of the neutral weights that appear in typical one-branch RGB-based approaches (LeNet,

AlexNet, VGGNet), as well as in our one-branch LAB filters approach.

The two-branch experiments were also made with the HSV and HSL color spaces, obtaining equivalent results (similar accuracies with fewer weights). However, the maximum accuracies are slightly below the ones obtained in the LAB space.

2.3.3 DenseNet-BC L40

As shown in table 2.3, required forward pass floating point operations have decreased by nearly 33% while maintaining the validation accuracy. In our tested hardware, we observed that the actual training time decreased by more than 33%, possibly because each path has smaller memory structures avoiding memory bottlenecks.

Branches	Color space	1st layer filters	Weights	Flops	Val. acc.
1	RGB	24(3ch)	176k	144M	92.0%
2	LAB	16(1ch)+8(2ch)	150k	97M	91.9%

Table 2.3: DenseNet-BC L40 results.

2.4 Conclusion

From our experiments, we conclude:

- Any CNN dealing with color images easily finds achromatic and chromatic patterns; the former are Gabor-like filters sensitive to edge/texture features, while the latter are single or opponent color features sensitive to colored areas and their boundaries.
- A CNN dealing only with the achromatic component of images (gray level) is capable of achieving a workable degree of performance (more than 80% in validation accuracy), but including chromatic information increases this performance enough (by almost 4%) to consider color as a relevant cue for object recognition.
- The chromatic filters tend to find a set of single colors (red, green, blue, yellow, etc.) and opponent-color contrasts (green-magenta, blue-yellow, red-cyan, etc., in several orientations) well spread within the color space.

- When dealing with 3-channel RGB filters, a CNN can adapt to the RGB high correlation and find pure achromatic, pure chromatic, and hybrid filters.
- When dealing with 3-channel LAB filters or alike (HSV, etc.), a CNN tends to focus on the achromatic channel or into the chromatic channels, letting the other subset of filter channels neutral (inactive); this is because LAB channels are uncorrelated; thus the learning process places the chromatic/achromatic patterns into the corresponding subset of weights.
- By splitting LAB filter values into two branches, one for L and another for AB, we can force a CNN to find prototypical sets of achromatic/chromatic filters but avoid the effect of inactive channels. In this way, the first layer of any CNN can be optimized by reducing more than 50% of the original number of weights. Our DenseNet implementation experienced a reduction in 33% of the required forward pass computation.
- According to our experiments, it seems that achromatic patterns are more relevant to object recognition than chromatic patterns; this is logical because object shape tends to be independent of object color (e.g., cars are painted in many different colors); however, some objects and backgrounds may render intrinsic colors (e.g., deers are usually brown, the sky and the sea are usually blue).

In essence, we have devised a modification of the first layer of a CNN into two branches, which optimizes the number of weights when dealing with a color encoding that separates achromatic from chromatic channels, such as LAB, HSL, etc. Although the proposed architecture does not increase the validation accuracy significantly, it eases the image classification learning of any CNN.

CHAPTER 3

Color-aware two-branch DCNN for efficient plant disease classification

In the previous chapter, we devised a modification of the first layer that splits the channels of an image encoded with HSL, LAB, or alike color space into two separate paths, one for the achromatic channel and another for the remaining chromatic channels. In this chapter, we modified an Inception V3 architecture to include one branch specific for achromatic data (L channel) and another for chromatic data (AB channels). This modification takes advantage of the decoupling of chromatic and achromatic information. Besides, splitting branches reduces the number of trainable parameters and computation load by up to 50% of the original figures using modified layers. We achieved a state-of-the-art classification accuracy of 99.48% on the PlantVillage dataset and 76.91% on the Cropped-PlantDoc dataset.

3.1 Introduction

Automating plant disease control is essential for early-stage symptom detection and continuous monitoring of crops. Such automation has a high impact on improving efficiency and productivity, especially in large fields Zambon et al. (2019). To automatically recognize plant leaf diseases, emerging AI technologies such as computer vision and deep convolutional neural networks (DCNNs) have been recently employed.

Initial studies used handcrafted features from leaf images Ngugi et al. (2021). Shallow classifier algorithms were proposed: the K-Nearest-Neighbors (KNN), Support Vector Machines (SVMs), decision trees, and shallow nonconvolutional neural networks Ngugi et al. (2021). Later, the trend switched to DCNN architectures capable of automatically extracting features and performing efficient classification Ngugi et al. (2021). Many CNN architectures, including LeNet Amara et al. (2017), CaffeNet Sladojevic et al. (2016), AlexNet Krizhevsky et al. (2012), GoogLeNet Ngugi et al. (2021); Mohanty et al. (2016); Ferentinos (2018), Inception V3 Ngugi et al. (2021); Ramcharan et al. (2017); Wang et al. (2017); Toda and Okura (2019); Maeda-Gutiérrez et al. (2020) and DenseNet Ngugi et al. (2021) have been applied to plant disease image classification.

Mohanty et al. (2016) worked with AlexNet and GoogLeNet models for the PlantVillage dataset classification. They trained both models from scratch and with transfer learning. They also experimented with feeding their models with RGB and grayscale images. They found better results feeding RGB images to both tested models. Their best result without transfer learning was 98.37%. G. and J. (2019) classified the PlantVillage dataset with 3 convolutional, 2 max poolings, and 2 dense layers achieving 96.46% of accuracy. Toda and Okura (2019) working with a trimmed Inception V3 showed that DCNNs can learn the colors and textures specific to plant leaf diseases resembling human-made classification.

Most previous architectures applied to plant leaf disease identification use the Red-Green-Blue (RGB) color values of input pixels. However, RGB components are highly correlated Pouli et al. (2013). Specifically, intensity variations induced by

illumination changes, edges, or texture modify the three RGB values by the same proportion.

In the previous chapter 2, we proposed a modification of the first layer that splits the channels of an image encoded with CIE Lab in two separate paths, one for the achromatic channel and another for the remaining chromatic channels. In this chapter, we investigate the influence of each branch on classification accuracy. For this aim, we created an Inception V3 Szegedy et al. (2016) based architecture that has two branches (paths) along the first three convolutional layers. One branch is fed from the L component, while the other is fed from the AB components. Furthermore, we test extreme cases by feeding our DCNN from only one of the two branches to check what our network can do with a single cue (chromatic or achromatic information) and without the other. In this respect, dealing with only the L channel makes our system compared with other methods that purely work on grayscale images.

For this work, we tested our DCNNs on the PlantVillage dataset Hughes and Salath'e (2015), which contains samples of 12 healthy crops and 26 crop diseases. We also tested our DCNNs on the Cropped-PlantDoc dataset Singh et al. (2020), which has 13 plant species and 27 classes of healthy and diseased crops.

The key contributions of this chapter to image-based plant leaf disease diagnostics can be summarized as follows:

- We present a feasible plant leaf image classification method based on an efficient DCNN architecture with separate branches dedicated to chromatic and achromatic information.
- We provide detailed performance analysis of several variants of our DCNN architecture, tested on the PlantVillage dataset Hughes and Salath'e (2015) and Cropped-PlantDoc dataset Singh et al. (2020).
- Our DCNN variants have achieved state-of-the-art plant leaf disease classification with 30% to 50% fewer filter weights and floating point computations along the first three convolutional layers.

The remainder of this chapter is structured as follows: section 3.2 presents and

discusses relevant work regarding DCNNs and image-based plant disease diagnostics. Section 3.3 presents our modified two-branch Inception V3 architecture. The results and discussion are given in sections 3.4 and 3.5. Section 3.6 summarizes the main conclusions.

3.2 Related Work

A number of machine learning methods have been proposed specifically for image-based plant disease diagnosis Ferentinos (2018); Sladojevic et al. (2016), including methods specifically designed for cucumbers Fujita et al. (2016), bananas Amara et al. (2017), cassavas Ramcharan et al. (2017), tomatoes Fuentes et al. (2017); Wang et al. (2017); Maeda-Gutiérrez et al. (2020) and wheat Johannes et al. (2017).

Ferentinos (2018) tested 5 existing architectures with a 58-class image dataset with healthy and sick plants: AlexNet, AlexNetOWTBn, GoogLeNet, Overfeat and VGG. Ferrentinos found test accuracies ranging from 99.06% with AlexNet to 99.48% with VGG. Despite the enormous difference in the number of trainable parameters in these architectures, the test accuracy was always above 99%.

Maeda-Gutiérrez et al. (2020) studied the application of 5 existing architectures, including: AlexNet, GoogleNet, Inception V3, ResNet-18 and ResNet-50, to tomato diseases. In this study, test accuracies were also found to be above 99%. Despite these excellent results, Ferentinos (2018) noted that there are problematic situations for images captured in the field, such as shading and leaves not centered in the image. Processing field images in the experiments drastically reduce the classification accuracy.

Chaudhary et al. (2012) studied plant disease spot segmentation in YCbCr, HSI, and CIE Lab color spaces. For all color channels, they experimentally found that feeding their segmentation model from the CIE Lab's A channel provides more accurate results than other channels. In their work, the A channel is a chromatic channel, providing better results than the achromatic L channel.

Amara et al. (2017) applied a plain LeNet architecture with 60x60 pixels images for banana leaf disease classification. Interestingly, they achieved an 85.94% test accuracy with grayscale images and a 92.88% test accuracy with RGB images. Mohanty et al. Mohanty et al. (2016) studied plant leaf diseases using grayscale and RGB color images processed with AlexNet and GoogLeNet (Inception V1) architectures. Their architectures were trained with the PlantVillage dataset Hughes and Salath'e (2015). Their best results were found with 80% of the image samples allocated for training while 20% of the samples were allocated for testing. The only architectural modifications are the number of classes and the input size set at 256x256 pixels for GoogLeNet. All experiments described at Mohanty et al. (2016) done with RGB images achieved higher accuracies than experiments done with grayscale images. Results obtained in Amara et al. (2017); Mohanty et al. (2016) indicate that chromatic information is essential for plant leaf disease classification.

Previous DCNN works from Ferrentinos Ferentinos (2018), Maeda et al. Maeda-Gutiérrez et al. (2020), Amara et al. Amara et al. (2017), and Mohanty et al. Mohanty et al. (2016) utilized off-the-shelf architectures with minor changes in the number of classes and the input layer.

G. and J. (2019) propose a DCNN with 3 convolutional layers, 2 max-pooling layers, and 2 dense layers trained with the PlantVillage dataset using data augmentation. After several batch size experiments and data augmentation experiments, they achieved a 96.46% classification accuracy. They distributed the dataset into 91%, 6%, and 3% of the samples for training, validation, and testing, respectively.

Toda and Okura (2019) proposal is based on the Inception V3 architecture Szegedy et al. (2016). The authors adjusted the last layer of the original network to fit in the 38 classes of the PlantVillage dataset. They distributed the dataset into 60%, 20%, and 20% of the samples for training, validation, and testing. Additionally, they performed an ablation study to determine the optimal number of mixed layers in the inception module, successively trimming the layers from the deepest to the shallowest. They found that with just the 6 former mixed layers, the network

provides very similar accuracy to the original network with 11 mixed layers (97.14% vs. 97.15%) while saving approximately 3/4 of the memory for storing the neuron weights (5.17 million vs. 21.88 million weights).

Ngugi et al. (2021) also worked with the PlantVillage dataset, but they used transfer learning. Since we train our networks from scratch, our results cannot be compared with their results. Transfer learning is a technique that uses the learned data from a related domain to improve the learning in a target domain Weiss et al. (2016). We prefer to focus on analyzing the impact of our architectural modifications and not care about the effect of improving the accuracy with the use of parameters transferred from another domain. Therefore, comparing our results with other works that apply transfer learning is unfair because they take advantage of pre-trained values. In contrast, we compare our test accuracies and F1 scores directly with those provided in Toda and Okura (2019), G. and J. (2019) and Mohanty et al. (2016) as those papers also use training from scratch.

Singh et al. (2020) created the Cropped-PlantDoc dataset, which has 13 plant species and 27 classes. Similar to the PlantVillage dataset, the original PlantDoc dataset includes pictures of individual leaves. However, those images also show complex backgrounds, and the area covered by the target leaves varies, which makes it a much more complex problem to classify than the PlantVillage images. To address this drawback, the authors manually crop the image regions containing target leaves. This provides conveniently framed leaves while significantly increasing the number of samples (approximately 9K) because they may extract several samples from each original PlantDoc image (approximately 2.6K).

3.3 Methodology

Figure 3.1 shows two designs of CNNs that analyze RGB pictures of plant leaves for plant disease classification. The design on the left corresponds to Toda & Okura's proposal, which we have chosen as our reference baseline model.

The design shown at the right of figure 3.1 corresponds to our proposal, which

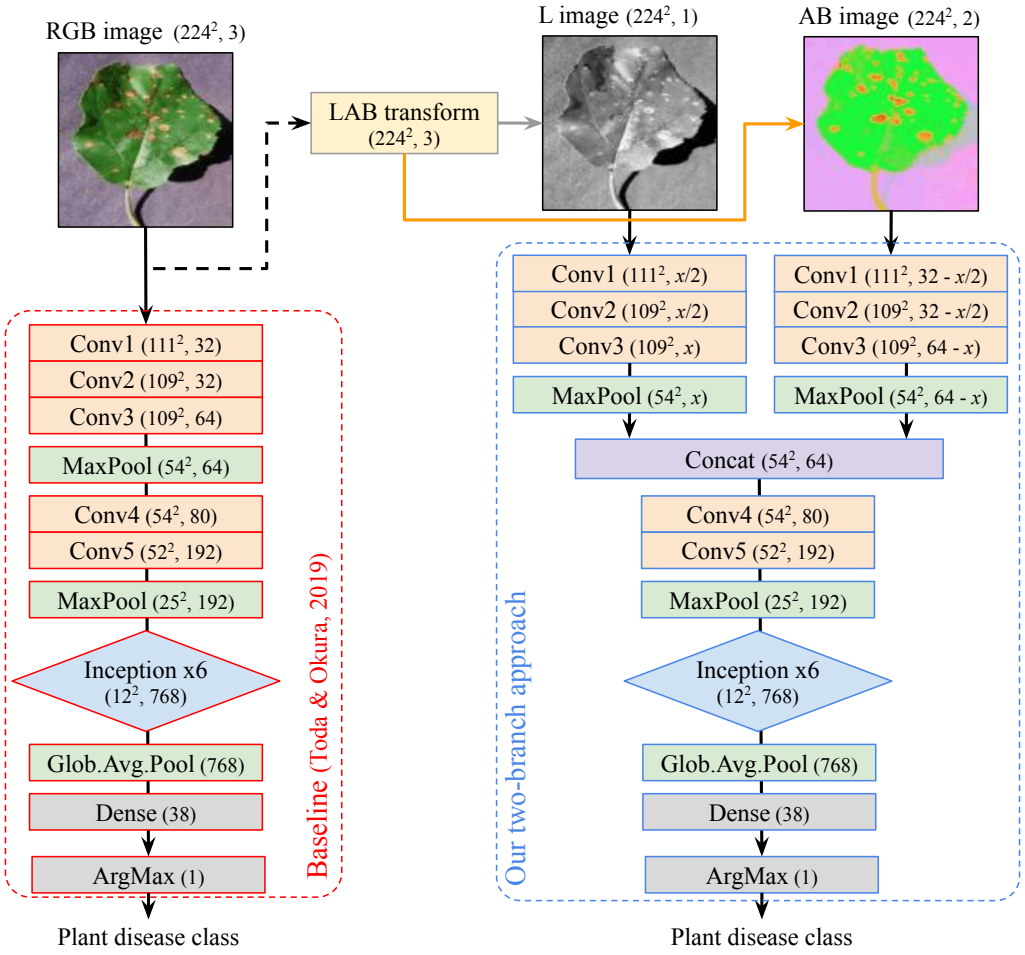

Figure 3.1: Graphical representation of worked network architectures: at the left, the Toda & Okura's single-branch (baseline) approach fed from an RGB image; at the right, our two-branch approach fed from L+AB images. Expressions containing x define a varying number of filters in L and AB branches. When $x/2$ is not an integer number, we use the floor function to round it.

splits the first three convolutional layers of the baseline into two branches, one for the L channel and another for the AB channels, computed from the input RGB image. Then, the output from each branch is concatenated to follow the rest of the network as in the baseline.

Another relevant remark is that we use a hyperparameter x to determine the distribution of the original number of filters among the L and AB branches. This allows us to determine the optimal contribution of each branch to the classification task. In the original Inception V3 implementation, the first 3 convolutional layers

have 32, 32, and 64 filters, respectively. We have mainly checked three variants for our proposal, named after the percentage of filters dedicated to achromatic (L) and chromatic (AB) branches: 20%L+80%AB, 50%L+50%AB, and 80%L+20%AB. For these variants, the value of x is set to 13, 32 and 51, respectively. Thus, the number of $L|AB$ filters in the first two layers will be 6|26, 16|16, and 26|6, respectively, for each variant. In the third layer, the number of $L|AB$ filters will be 13|51, 32|32, and 51|13, respectively, for each variant.

To compare our proposal with the baseline reasonably, we have imposed that the sum of the filters of the two branches in each layer must be the same as in the Inception V3 design. However, our filters carry a fraction of the original number of weights (from 1/3 to 2/3).

model	1st layer weights	2nd layer weights	3rd layer weights
baseline	0.8k	9k	18k
20%L + 80%AB	0.5k	6k	13k
50%L + 50%AB	0.4k	**5k**	**9k**
80%L + 20%AB	**0.3k**	6k	13k

Table 3.1: Number of weights for each of the first 3 convolutional layers, for baseline and our variants.

model	1st layer flops	2nd layer flops	3rd layer flops
baseline	21M	227M	453M
20%L + 80%AB	12M	158M	315M
50%L + 50%AB	10M	**113M**	**226M**
80%L + 20%AB	**8M**	158M	315M

Table 3.2: Number of required forward pass floating point operations for each of the first 3 convolutional layers, for baseline and our variants.

The implementation details of our approach are generally based on Toda and Okura (2019) which is Inception V3 based (see figure 3.1). Each convolutional layer is composed by a 2D convolution, batch normalization, and a ReLU as the activation function. All convolutional filters from Conv1 to Conv5 are 3×3 except for Conv4, which is 1×1. In Conv1, there is a stride of 2. All convolutional layers from Conv1 to Conv5 do not have padding except for Conv4, which is zeroed padded by

1px. We do not use data augmentation as we study the net effect of the proposed architecture (L/AB separate branches) on the accuracy, beyond extra refinements (data augmentation, transfer learning) that may lead to some degree of improvement but not due to the proposed architecture. The training data is shuffled before each epoch. The optimization method is stochastic gradient descent. The loss function is the categorical cross entropy function. The batch size is 32. The test accuracy is obtained with the parameters from the epoch with the highest validation accuracy. All models have been trained from scratch without transfer learning.

All experiments were implemented with K-CAI Schuler (2021) and Keras Chollet et al. (2015) on top of Tensorflow 2 Abadi et al. (2015), with various underlying hardware configurations including NVIDIA 1070, 1080, K80, T4 and V100 video cards. Our virtual machines have up to 64GB of RAM. The implementation details of our approach are strongly based on the reference paper Toda and Okura (2019). Each convolutional layer comprises a 2D convolution, a batch normalization, and a ReLU activation function. All convolutional filters from Conv1 to Conv5 are of the size 3×3 except for Conv4, which is 1×1. The optimization method is stochastic gradient descent, and the loss function is weighted categorical cross entropy to compensate for an unbalanced number of samples among classes. The batch size is 32, and we store the weights that obtain the best validation accuracy in 30 epochs. We trained all models from scratch. The noise injection module has not been used for training since this module is only intended to verify the reliability of the models under controlled perturbation of the test images.

For the PlantVillage dataset, we trained all DCNNs for 30 epochs with a constant learning rate of 0.01. We split the PlantVillage Dataset into 60% of samples for training, 20% for validation, and 20% for testing. After a random dataset is split into the training, validation, and testing subsets, this splitting will be used in all experiments to ensure that results from different experiments are not affected by the sample splitting. For this dataset, we weight the loss function according to the number of samples per class to give the same relevance to each class. This gives similar classification accuracy across all classes. For the Cropped-PlantDoc dataset,

we trained all DCNNs for 240 epochs, starting with a learning rate of 0.01 and decaying 1% per epoch. We split the Cropped-PlantDoc dataset into 65% of the samples for training, 15% for validation, and 20% for testing.

Our source code for these experiments and their raw result files is publicly available at https://github.com/joaopauloschuler/two-branch-plant-disease/.

3.4 Results

We have assessed the performance of several models: the original Toda et al. single-branch architecture for RGB images, which we refer to as the baseline, and three variants of our two-branch architecture for L—AB channels. For these experiments, we computed the classification accuracy for the testing subset, and the multiclass F1 score Rijsbergen (1979).

Table 3.3 collects the results for the PlantVillage dataset. It shows that our two-branch 20%L+80%AB variant provides the best test accuracy and F1 score. It renders a modest but clear improvement (1.11% in accuracy and 0.87% in F1) over the best pre-existing model, Mohanty's GoogleLeNet. This 20%L+80%AB variant is slightly better than the other two-branch variants (up to 0.4% test accuracy). The baseline underperforms our worst variant (two-branch 80%L+20%AB) with a gap of almost 2 percentage points in test accuracy.

The worst models (Mohanty's AlexNet and Mohanty's GoogLeNet) are both fed Gray images. The other RGB-based architectures achieve test accuracies ranging from 96.46% to 98.37%. In short, we can sort these results into 3 groups: LAB-fed two-branch models (highest accuracy), RGB-fed models (middle accuracy), and gray-level fed models (lowest accuracy). Our worst two-branch variant has a 0.71% higher accuracy and a 0.32 higher F1 score than the best performing RGB implementation.

We have also tested our DCNNs on the Cropped-PlantDoc dataset Singh et al. (2020). On this dataset, we reproduced the experiment from Toda et al. regarding trimming the number of Inception V3 mixed layers using our 20%L+80%AB variant.

author	architecture	color space	parameters	accuracy	F1
Schuler	**20%L+80%AB**	**L—AB**	**5M**	**99.48%**	**0.9923**
Schuler	50%L+50%AB	L—AB	5M	99.11%	0.9866
Schuler	80%L+20%AB	L—AB	5M	99.08%	0.9867
Mohanty	GoogLeNet	RGB	5M	98.37%	0.9836
Mohanty	AlexNet	RGB	60M	97.82%	0.9782
Toda	Inception V3	RGB	5M	97.15%	0.9720
Geetharamani	9 layers CNN	RGB	0.2M	96.46%	0.9815
Mohanty	GoogLeNet	Gray	5M	96.21%	0.9621
Mohanty	AlexNet	Gray	60M	94.52%	0.9449

Table 3.3: Test accuracy and F1 score of several DCNN models on PlantVillage dataset classification. The results extracted from other papers are those obtained without transfer learning (not their best ones) for a fair model comparison with our results since we do not use transfer learning.

Table 3.4 shows the obtained results. In this experiment, we concluded that the ideal number of mixed layers is 6.

architecture	color space	mixed layers	max. val. accuracy	test accuracy
20%L + 80%AB	L—AB	1	74.12%	72.50%
20%L + 80%AB	L—AB	2	77.27%	76.97%
20%L + 80%AB	L—AB	4	77.19%	74.68%
20%L + 80%AB	**L—AB**	**6**	**78.77%**	**77.08%**
20%L + 80%AB	L—AB	8	77.12%	73.90%
20%L + 80%AB	L—AB	10	75.62%	73.84%
20%L + 80%AB	L—AB	11	73.14%	74.23%

Table 3.4: Max validation and test accuracies when trimming the number of mixed layers, trained on Cropped-PlantDoc dataset.

Once we decided to use just 6 mixed layers, we trained all our variants on the Cropped-PlantDoc dataset from scratch again. Our next results outperformed all previous models from Singh et al., as shown in table 3.5. Interestingly, we have not used transfer learning; while Singh et al. used transfer learning, we could not find any other proposal using this dataset to train from scratch. It is also vital to note that our variants have less than 10% of the trainable parameters used by Singh et al.'s best model while exceeding its test accuracy by more than 6 percentage points.

In table 3.5, we have also included two extreme variants of our model: 0%L + 100%AB and 100%L + 0%AB. In those variants, the DCNN is fed pure chromatic or achromatic information; hence, the DCNN operates with a single branch. The

author	architecture	color space	parameters	accuracy	F1
Schuler	0%L + 100%AB	L—AB	5M	71.55%	0.71
Schuler	20%L + 80%AB	L—AB	5M	76.58%	0.76
Schuler	**50%L + 50%AB**	**L—AB**	**5M**	**76.91%**	**0.76**
Schuler	80%L + 20%AB	L—AB	5M	75.85%	0.75
Schuler	100%L + 0%AB	L—AB	5M	64.67%	0.66
Singh	InceptionResNet V2	RGB	55M	70.53%	0.70
Singh	InceptionV3	RGB	22M	62.06%	0.61
Singh	VGG16	RGB	138M	60.41%	0.60

Table 3.5: Test accuracy and F1 score with the Cropped-PlantDoc dataset.

test accuracy from these experiments indicates that color is more important than gray-level information for classifying samples from this dataset. However, achromatic information also plays a role in plant disease classification since our best result has been obtained with a combination of both image cues in the same proportion.

3.5 Discussion

Typical first layer filters of DCNNs specialize in gray level features (Gabor-like filters) or in color-opponent filters, as in Krizhevsky et al. (2012) and Zeiler (2014). In our design, our L filters only need one channel to learn the spatial patterns defined by the gray-level component. In addition, RGB filters replicate the same weights in their three channels to represent those gray-level patterns. Hence, L filters save 2/3 of the weights used by RGB filters. Similarly, for the case of color-opponent filters, our AB filters learn them using only two chromatic channels, while regular RGB filters employ three channels for the same task. Therefore, AB filters save 1/3 of the weights. Our design saves from 1/3 to 1/2 of the weights in the first three layers. Also, it achieves similar savings in the computational floating point operations for carrying those convolutions, as shown in tables 3.1 and 3.2.

Table 3.3 shows that the two-branch approach 20%L + 80%AB is slightly better than the classical RGB single-branch approach. This indicates that separating filters for achromatic-chromatic features enhances the classification ability of DCNNs. However, the difference in classification accuracy concerning the baseline architecture

is slight because the DCNN optimization procedure can decorrelate the features encoded in the RGB channels. With a sufficient number of training epochs, an RGB single-branch DCNN will identify filters sensitive to lightness (Gabor-like grayscale filters) and other filters sensitive to color contrasts (see examples of RGB filters in Krizhevsky et al. (2012) and Zeiler (2014)). Nevertheless, our two-branch methodology obtains similar accuracy with 30% to 50% fewer filter weights and floating point operations along the first 3 convolutional layers. This proves that the extra weights encoded in the first three layers of the RGB single-branch approach are redundant. All grayscale experiments have lower classification accuracy than their RGB counterparts, corroborating that chromatic information is essential for plant leaf disease identification.

The results in table 3.4 show that a trimmed version of Inception V3, with less than 25% of the original parameters (5 million vs. 22 million), is capable of performing significantly better (5% in test accuracy) than the full-fledged version. We think that the extra number of parameters may become a drawback for training due to overfitting.

Finally, table 3.5 corroborates the advantage of our proposal. The results of these experiments show more significant differences than the results in table 3.3 because most of the methods obtained truly high test accuracy on the PlantVillage dataset, leaving little room for improvement. Since the Cropped-PlantDoc dataset is much more difficult to classify, our simple but effective methodology shows its advantage over single-branch RGB-based approaches.

3.6 Conclusion

In this chapter, we proposed a two-branch DCNN for plant disease classification, where the first three convolutional layers specialize in learning chromatic and achromatic features from the CIE Lab color space.

The experiments conducted empirically prove that our approach can perform better than the classic one-branch RGB images fed DCNN while saving a portion

of learnable parameters and floating point operations, reducing the numbers from 1/3 to 1/2 in those initial three layers. This is feasible because the RGB channels are highly correlated; working in a decor-related color space avoids redundant filter weights.

Concerning the optimal distribution of filters among achromatic and chromatic branches for plant disease classification, our experiments show that approximately 50% to 80% of the filters should enter the chromatic branch. This indicates that color is essential for this task.

CHAPTER 4

Noise Resistant Plant Leaf Disease Classification

In chapter 2, the two-branch approach is introduced. In chapter 3, we modified an Inception V3 to process CIE Lab encoded images achieving state-of-the-art classification accuracy of 99.48% on the PlantVillage dataset. This chapter shows that this two-branch architecture provides better classification reliability when perturbing the original RGB images with several types of noise (salt and pepper, blurring, motion blurring, and occlusions). These types of noise simulate common image variability found in the natural environment. We hypothesize that the filters in the AB branch provide better resistance to these types of variability due to their relatively low frequency in the image-space domain.

4.1 Introduction

Plant leaf images taken in the field and away from controlled laboratory conditions frequently suffer from blurring, motion blurring, occlusion, and illumination variations. Automated detection systems frequently suffer from these common adverse effects. In this chapter, we show that the two-branches architecture described in the previous chapter provides more resistance to adverse effects such as blurring.

4.2 Methodology

In this chapter, we use the same two-branch architecture described in the previous chapter and shown in figure 3.1. In addition to this architecture, to verify the reliability of this architecture, we have included one module for noise injection. This allows us to perturb the original RGB images with different types of artifacts and varying degrees of severity of those artifacts. It must be observed that the noise injection is previous to the RGB-to-LAB transformation.

As explained in chapter 2, RGB channels are highly correlated among each other Pouli et al. (2013) in the sense that shading and shadows render a set of different RGB values from the intrinsic color(s) of a surface. Specifically, intensity variations induced by illumination variation, edges, and texture modify the three RGB values simultaneously. We also have mentioned that transforming RGB channels into some sort of achromatic-chromatic space, like CIE Lab, effectively isolates the gray-level features in the L channel and the color-related features in the AB channels. Due to our two-branch architectures, we are forcing the filters in each branch to learn features related to the nature of each cue. Therefore, we expect that L filters will focus on intrinsic shape, damaged leaf areas, etc., while the AB filters will focus on lesions, general color of the leaf, etc. We hypothesize that noise will not affect chromatic and achromatic branches with the same severity as the noise is decoupled into achromatic and chromatic information and loaded into its branch.

model	weights (Saving)	flops (Saving)
baseline	28512	701M
20%L + 80%AB	19746 (31%)	485M (31%)
50%L + 50%AB	**14256 (50%)**	**350M (50%)**
80%L + 20%AB	19566 (31%)	481M (31%)

Table 4.1: Weights and required forward pass floating point operations along the first 3 convolutional layers in baseline and our variants.

4.3 Results

Figure 4.1 shows the evolution of test accuracy in the studied models, baseline, two-branch 20%L-80%AB, 50%L-50%AB, and 80%L-20%AB, for different types of noise and a range of noise amount.

In Salt and Pepper experiments, the range of noise indicates the percentage of input image pixels that have been changed to either white or black pixels (see Fig. 4.2 for an example). This type of noise simulates spuriously saturated values in the input signal. The corresponding plot depicts the 20%L-80%AB variant as the most reliable when the percentage of noisy pixels is above 3% as the classification accuracy is up to 10% more accurate than the baseline. Nevertheless, the baseline performs better than the other two branched models in the range of noise used for these experiments.

In Blur experiments, a Gaussian distribution of a given sigma in image space coordinates (distance in pixels) is convolved with the input RGB image values producing the typical blurring effect (check Fig. 4.2). This type of noise simulates unfocused snapshots or dirty lenses. In the corresponding plot, our 20%L-80%AB variant proves the most reliable under the tested range of sigmas. From $\sigma = 1.25$ to $\sigma = 1.75$, this best model overcomes the baseline by 10% of test accuracy. Moreover, the 50%L-50%AB variant also overcomes the baseline, albeit slightly different.

Motion blur is similar to blur (also check Fig. 4.2), but instead of a Gaussian distribution, we use a sparse matrix of a given size with all cells equal to zero except for one line of cells, which is filled with ones divided by the number of cells in that line. By convolving the image pixel values with such a matrix (kernel), it is possible to simulate the blurring due to sudden camera shifts. The direction of movement is

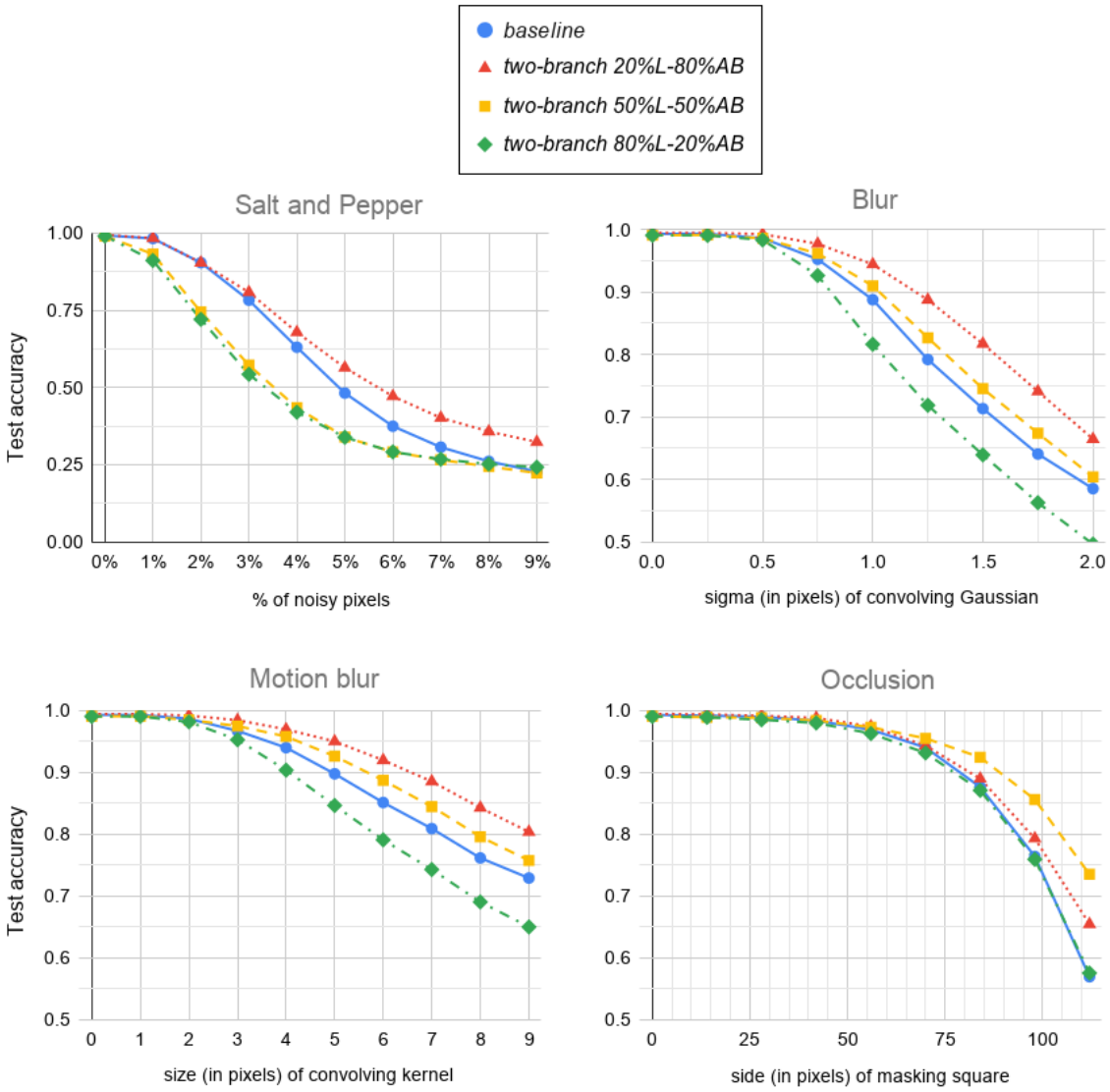

Figure 4.1: Result plots showing the test accuracy evolution of four approaches under a range of perturbation with four types of noise.

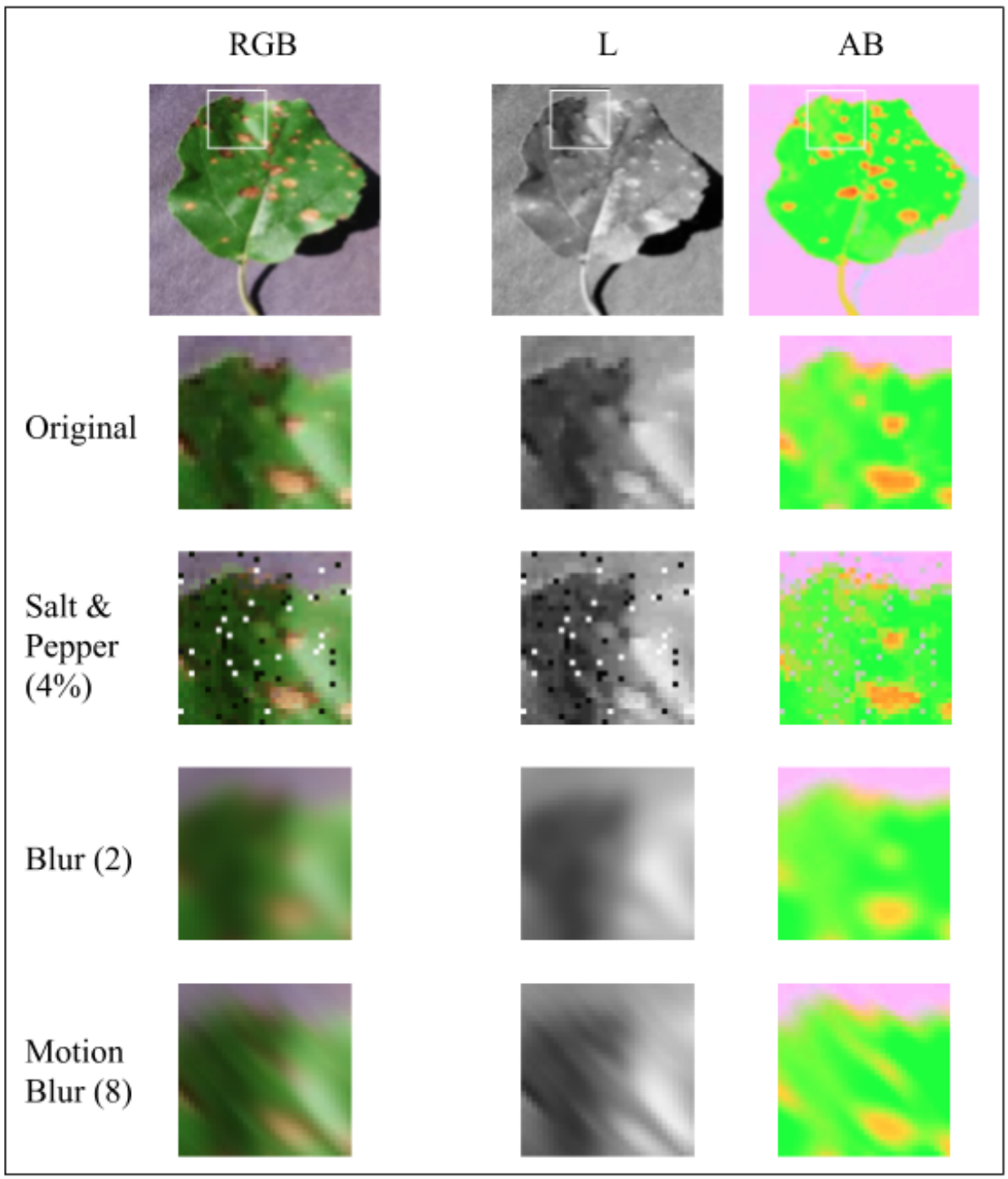

Figure 4.2: Noise injection in a portion of a test image (Apple Black Rot num.5), in RGB, L, and AB spaces: Salt & Pepper noise in 4% of the image pixels; Blur by convolving a Gaussian bell with $\sigma = 2$ pixels; Motion Blur in up-left direction with 8 pixels of kernel width.

parallel to the line of cells different from zero. The extent of movement is equivalent to the length of that line. The corresponding plot depicts similar behavior to the blurring plot. However, it is necessary to use a 9 pixels-side kernel to degrade the test accuracy of the 20%L-80%AB variant as much as with a $\sigma = 1.5$ in the blurring experiment.

Occlusion is performed by overlapping a square of gray pixels of a given size in a random image position. This type of noise simulates the occlusion of the target leaf by other non-interesting objects such as tree branches, fruits, etc. For these experiments, the model that renders the best reliability in the corresponding plot is our 50%L-50%AB variant with a remarkable difference of 5% above the second best model, the 20%L-80%AB variant, which in turn is also 5% above baseline and the 80%L-20%AB variant when the side of the masking square is beyond 100 pixels.

4.4 Discussion

All results are highly determined by the fact that the leaf shape and their lesions are less varying in AB channels than in RGB and L channels, as can be seen in the example of Figure 4.2. In other words, the leaf representation in AB channels renders broad areas of similar colors. This low-frequency nature of the AB channels makes the color-trained filters inherently take into account a wider field of view. Therefore, more erroneous pixels are needed to mislead the classification. In contrast, the same leaf surface renders more frequent variations in RGB channels which provokes that their trained filters will have a smaller field of view. Specifically, high-frequency noise affects more gray-level filters, which are the ones projected into the L channel. These observations may explain why focusing 80% of the filters on the AB branch provides the best results in the presence of most types of noise.

For salt and pepper noise, the effect of spurious pixels in AB channels is noticeable, but the larger field of view of corresponding filters allows to overcome those perturbed values. On the other hand, the field of view of L and RGB filters is closer to the area of each erroneous pixel. However, the baseline is more reliable

than 50%L-50%AB and 80%L-20%AB configurations because its filters can better treat the spurious changes in the 3D RGB space than the combination of the split L and AB filters.

In contrast to salt and pepper, blurring is a perturbation of low-frequency nature. Despite this fundamental difference, our 20%L-80%AB configuration becomes again the most reliable. In this case, the smoothing of pixel values degrades more the features encoded in the L and RGB channels than the features encoded in the AB channels. The 50%L-50%AB configuration is also more potent than the baseline. Regarding motion blurring, the 20%L-80%AB and 50%L-50%AB configurations are again the most reliable.

For the occlusions experiment, the 50%L-50%AB and 20%L-80%AB variants are the most resilient, especially for mask sizes above 1/4 of the total image area. Again, the reasoning for this effect is that a prominent occlusion in the AB image removes less relevant details than the same occlusion in L and RGB images, as the key features in AB channels are wider in image space than in L or RGB channels.

4.5 Conclusion

In this chapter, we verified that our two-branch CNN for plant disease classification proposed in the previous chapter is more reliable in front of different noise sources than a typical CNN based on RGB. Besides classifying images with fewer weights than the baseline model, our experiments show that our 20%L-80%AB and 50%L-50%AB models better classify input images under salt and pepper, blurring, motion blurring, and occlusion by margins up to 10%.

With regards to the optimal distribution of filters among achromatic and chromatic branches, our experiments show that about 80% of the filters should go into the chromatic branch to provide maximum reliability in front of different sources of noise. The reason behind this conclusion is based on the fact that color filters have a wider field of view than lightness or RGB filters. Another reason is the color cue portrays highly relevant features for plant disease classification.

CHAPTER 5

Grouped Pointwise Convolutions Reduce Parameters in Convolutional Neural Networks

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 parallel branching. 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 when trained from scratch, DCNNs with the proposed technique 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.

5.1 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 convolutions. Unlike (spatial) 3x3 convolutional filters, these filters do not consider surrounding positions.

This chapter 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 fewer floating point operations. For instance, in the context of plant disease classification, the Cropped-PlantDoc dataset Singh et al. (2020) 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. This chapter 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 state-of-the-art classification accuracy on any image classification dataset is out of the scope of this chapter, as we focus on DCNN optimization.

This is how this chapter is organized: Section 5.2 introduces and examines relevant work, parameter-efficient DCNNs, and the datasets used in this chapter. The proposed pointwise convolution optimization is explained in Section 5.3. Sections 5.4 and 5.5 provide results and discussion, respectively. The chapter is summarized in Section 5.6.

5.2 Related work

In 2013, Min Lin et al. introduced the Network in Network architecture (NiN)Lin et al. (2014). This architecture contains 3 spatial convolutional layers with 192 filters, 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. When their work was released, they attained state-of-the-art classification accuracies in the CIFAR-10 and CIFAR-100 datasets Krizhevsky (2009).

In 2016, ResNet was introduced. Resnet He et al. (2016b) stacks up to 152 layers with similar topology. Inspired on VGG Simonyan and Zisserman (2015), 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 to skip connections every 2 convolutional layers.

Ioannou et al. (2017) 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, their study did not attempt to separate the 1x1 pointwise convolutional layers. Ioannou et al. also worked on an optimized ResNet-50 variant by replacing the original spatial convolutions with 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 the ImageNet dataset. They observed that it is unlikely that every filter depends on all output channels coming from the previous layer. This observation is fundamental to this thesis, as we use it to 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 Xie et al. (2017) was introduced. However, ResNeXt replaces the spatial convolutions with 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 Russakovsky et al. (2015) 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. (2017) 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 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, Howard et al. (2019) an improved version of MobileNet was introduced, named V3, still based on depthwise and pointwise convolutions, but pointwise convolutions remained ungrouped.

Zhang et al. (2017) 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 Zhang et al. (2017) and our technique is that we target replacing pointwise convolutions.

In Tan and Le (2019), 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 and MobileNets, which allows for a significant decrease in the number of parameters and floating-point operations, which we have taken advantage of in this thesis.

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

- The **Oxford-IIIT Pet dataset** Parkhi et al. (2012): 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** Krizhevsky (2009): as described in chapter 2, 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 the class refers to. The objects are sometimes partially obscured or viewed from an unexpected angle.

- The **CIFAR-100 dataset** Krizhevsky (2009): 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** Singh et al. (2020): as described in chapter 3, 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. In this dataset, images have heterogeneous backgrounds, and the leaves significantly differ in size.

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) Zhou et al. (2016). The class activation map method finds image regions used by a CNN to classify an image. Regions relevant for the classification are shown from red (more relevant) to blue (less relevant). This method can be used when the last two 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).

5.3 Methodology

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. 5.1, where F_i is the total number of filters as found in the original monolithic pointwise convolution layer:

$$P_i = C_{i-1} \cdot F_i \tag{5.1}$$

We propose a method to make pointwise convolutions parameter-efficient. Figure 5.1 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 its own set of input channels, preventing channels connected to different groups from being combined. 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 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 does not raise 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, which 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 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. 5.2:

$$P_i = (C_{i-1} \cdot F_i)/N_i \tag{5.2}$$

Eq. 5.2 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 two channels from the same group are not placed together.

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 in $N_i = 64$ 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 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 initially add up to 2 convolutions where we initially find one. Contrary to the work of Ting Zhang et. al. Zhang et al. (2017), we do not have special convolutions; we sum the outputs of both grouped convolutions via a skip connection. This added skip connection counterbalances 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 varies 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 to test our ideas on already efficient architectures. In the original architectures, when the last convolutional layer has a 1x1xC activation map as input shape, it is left unmodified as it behaves as a dense layer for the final classification. For the kEffNet-B0, 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 Schuler (2021); Chollet et al. (2015); Abadi et al. (2015) and RMSProp optimizer. All 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 for the limited number of images in these datasets, we trained the DCNNs for over 50 epochs. It is worth noting that the number of epochs is a multiple of our learning rate cycle, which is 25. In this chapter, we did not employ transfer learning as our main objective is to

assess the learning capacity of parameter-efficient DCNN models.

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

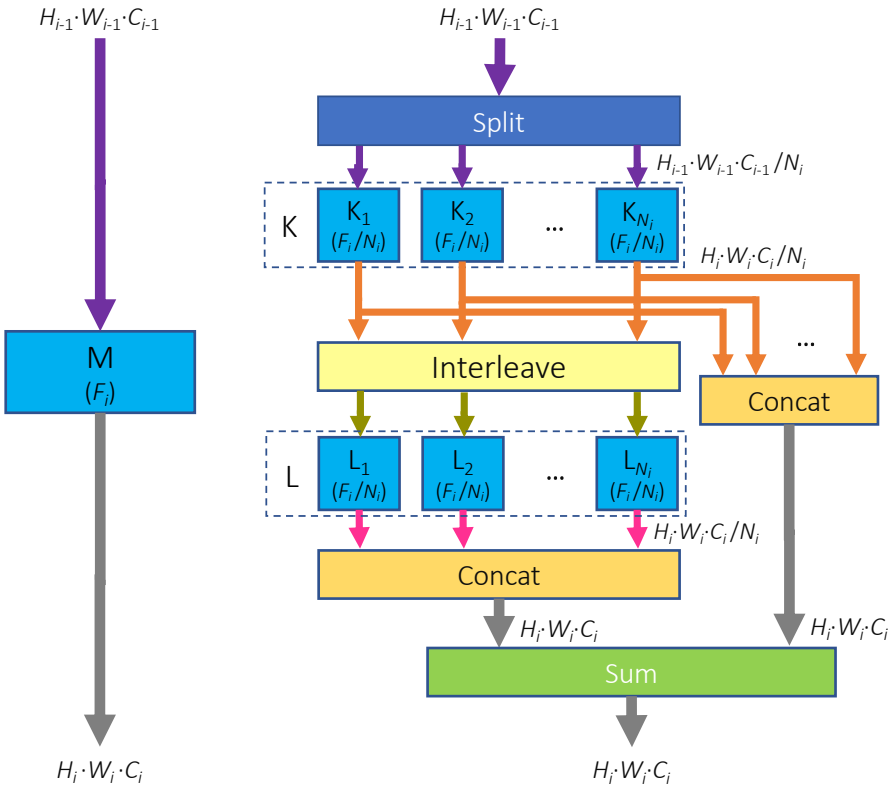

Figure 5.1: Diagram of the proposed pointwise convolution optimization. (Left) a classic monolithic layer M with F_i pointwise filters. (Right) our proposed replacement for M. It comprises two grouped pointwise convolutional layers (K and L) with N_i groups, where each group consists of F_i/N_i filters. H, W, and C represent the channels' height, width, and number. 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.

5.4 Results

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

5.4.1 Analyzing Classification Accuracy

Table 5.1 compares test accuracies, the number of trainable parameters, the 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. For 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 the smallest number of parameters outperforming 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 small architecture outperforming 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.

Table 5.1: 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%**

In table 5.1, 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 differs 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 parameter saving doesn't necessarily result in a proportional calculation saving.

Table 5.2: 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**%

The two best performing variants kEffNet-B0 and kMobileNet V3 in table 5.1 were retested with CIFAR-100, Cropped PlantDoc and Oxford-IIIT Pet datasets as per tables 5.2, 5.3 and 5.4 respectively.

In our Cropped PlantDoc experimentation shown in the table 5.3, 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 higher accuracy than its baseline by a margin of 0.5%.

In our Oxford-IIIT Pet dataset experimentation shown in table 5.4, we obtained highest accuracy with kEffNet-B0 16ch (64.56%) followed by kMobileNet V3 Large 32ch (63.09%). We speculate that this result results from overfitting due to a small

Table 5.3: 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%**

training dataset. In all other tested datasets, the 32ch variant achieves higher accuracy than its 16ch-related variant.

Table 5.4: 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%**

5.4.2 Class Activation Maps

Figure 5.2 shows CAMs made with the Oxford-IIIT Pet dataset. Our kEffNet-B0 focuses on the face, ears, and the top of the head. In turn, the EfficientNet-B0 baseline has its focus more frequently in the background.

Figure 5.3 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.

5.5 Discussion

Indeed, the best-performing optimized architectures are based on EfficientNet, MobileNet, and MobileNet V3. These 3 architectures share a characteristic in common that explains why our optimization fits well: in the original architectures,

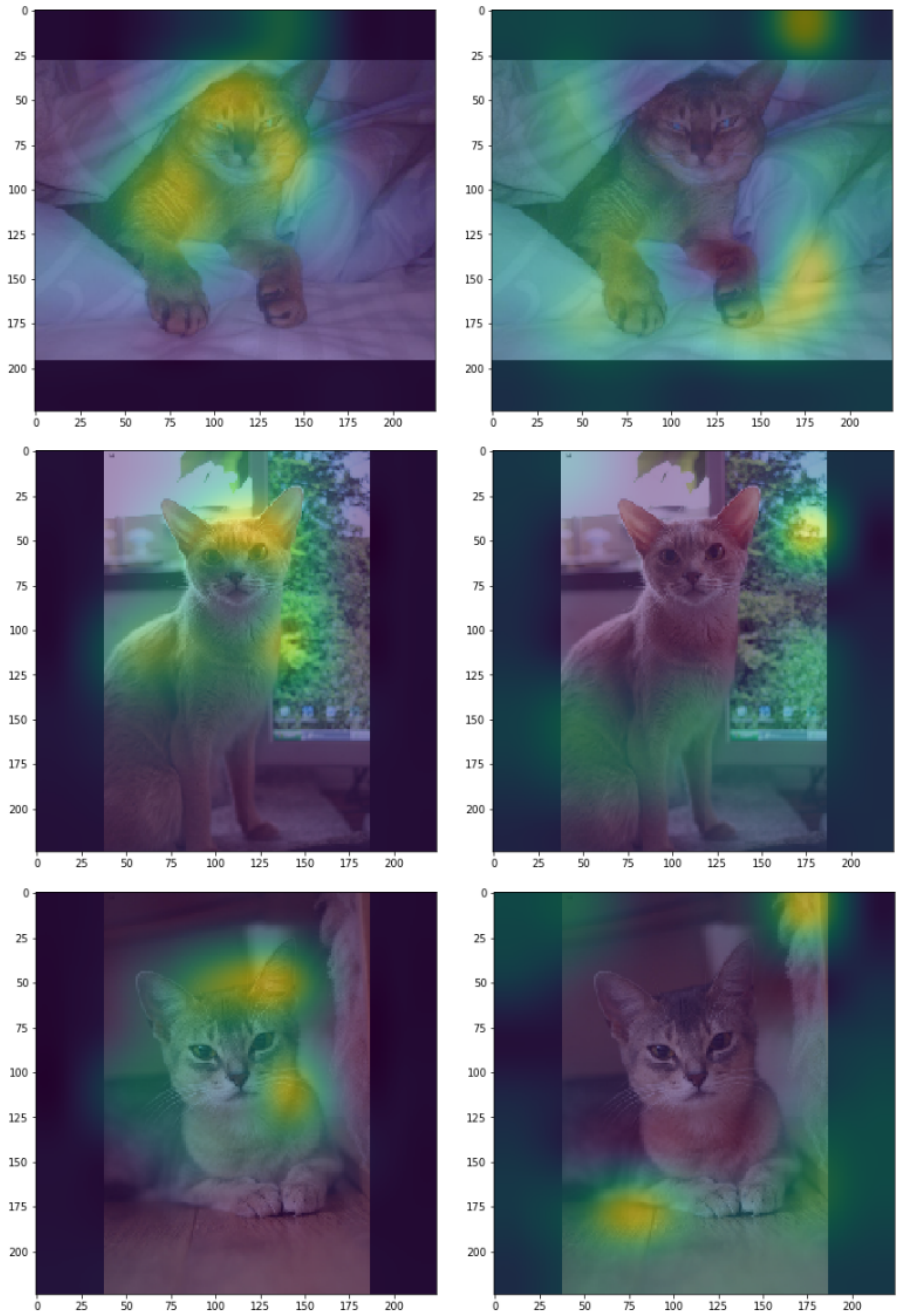

Figure 5.2: CAMs for images taken from the Oxford-IIIT Pet dataset. (left) CAMs produced by kEffNet-B0 with the proposed pointwise optimization technique. (right) CAMs produced by the EfficientNet-B0 baseline.

the main convolutions are parameter efficient depthwise convolutions and parameter inefficient pointwise convolutions. Given that our replacement is explicitly designed to reduce excess connections in pointwise convolutions, we apply our optimization at the precise, weak point of the original architectures.

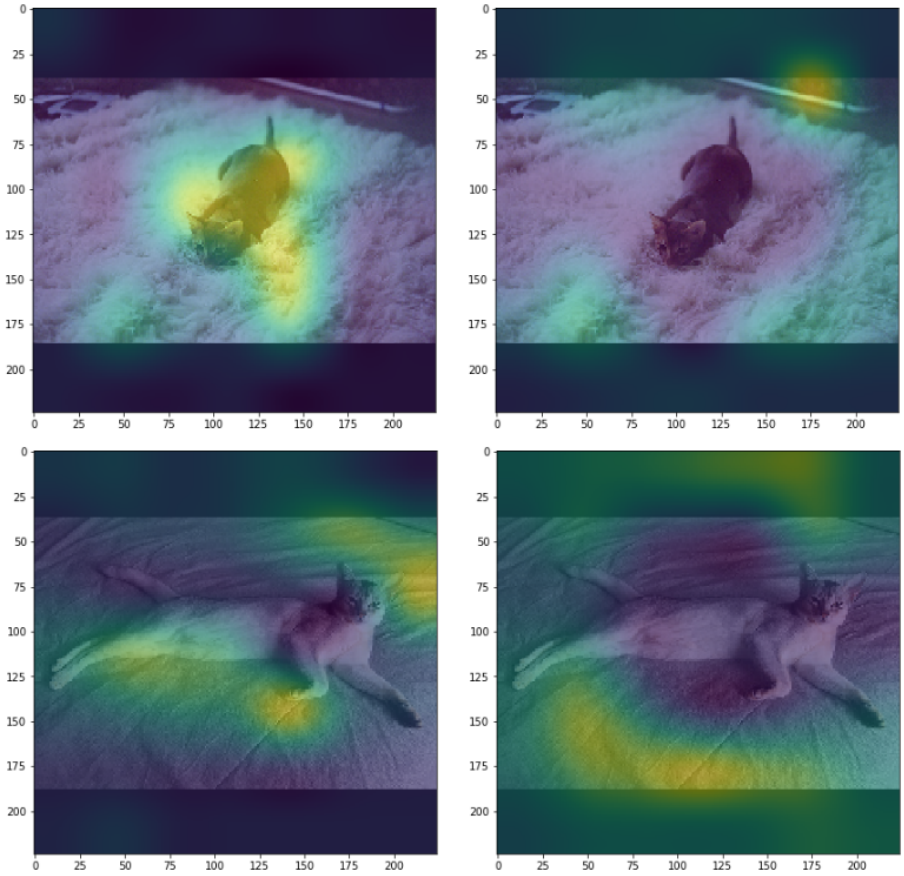

Figure 5.3: CAMs showing image samples which architectures do not focus on the cat. (left) CAMs produced by kEffNet-B0 with the proposed pointwise optimization technique. (right) CAMs produced by the EfficientNet-B0 baseline.

The variants derived from DenseNet-BC L100 underperformed the original model. In DenseNet-BC, most pointwise convolutions are followed by standard convolutions 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.

The proposed pointwise replacement does not directly explain why we obtained better classification accuracy with Cropped-PlantDoc or 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

technique 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.

5.6 Conclusion

This chapter 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. For 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 accuracy than the baseline when trained from scratch. In 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 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%). These results indicate that our optimization works better on architectures with 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, and 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.

CHAPTER 6

Grouped Pointwise Convolution Refinement

In the previous chapter, we showed a subnetwork that replaces pointwise convolutions with significantly fewer parameters and floating-point computations while maintaining the learning capacity. Our subnetwork utilizes grouped pointwise convolutions, in which each group processes a fraction of the input channels. In the present chapter, we refine the previous algorithm so that groups can have several filters to cope with non-divisible numbers of input channels, output channels, and groups. In this case, our previous method overlooked and did not replace the original pointwise convolutions. Thus, the new method further reduces the number of floating-point computations (11%) and trainable parameters (10%) achieved by the previous method. We tested our optimization on an EfficientNet-B0 as a baseline architecture and made classification tests on the CIFAR-10, Colorectal Cancer

Histology, and Malaria datasets. For each dataset, our optimization saves 76%, 89%, and 91% of the number of trainable parameters of EfficientNet-B0, while keeping its test classification accuracy.

6.1 Introduction

In the previous chapter, we proposed replacing standard pointwise convolutions with a sub-architecture that contains two grouped pointwise convolutional layers (K and L), an interleaving layer that mixes channels from layer K before feeding layer L, and a summation at the end that sums the results from both convolutional layers. Our original method accepts a hyperparameter Ch, which denotes the number of input channels fed to each group of filters. Then, our method computes the number of groups of filters and filters per group according to the division of original input channels and filters by Ch. Our original method avoided substituting the layers where the divisions were not exact.

In this chapter, we propose an enhanced scheme to allow flexible computing of the number of groups, so the divisibility constraints do not have to be considered anymore. By applying our method to all pointwise convolutional layers of an EfficientNet-B0 architecture, we can reduce a considerable amount of resources (trainable parameters, floating-point computations) while maintaining the learning capacity.

This chapter is structured as follows: Section 6.2 details our improved solution for grouping pointwise convolutions while skipping the constraints of divisibility found in our previous method. Section 6.3 details the experiments carried out for testing our solution. Section 6.4 summarizes the conclusions and limitations of our proposal.

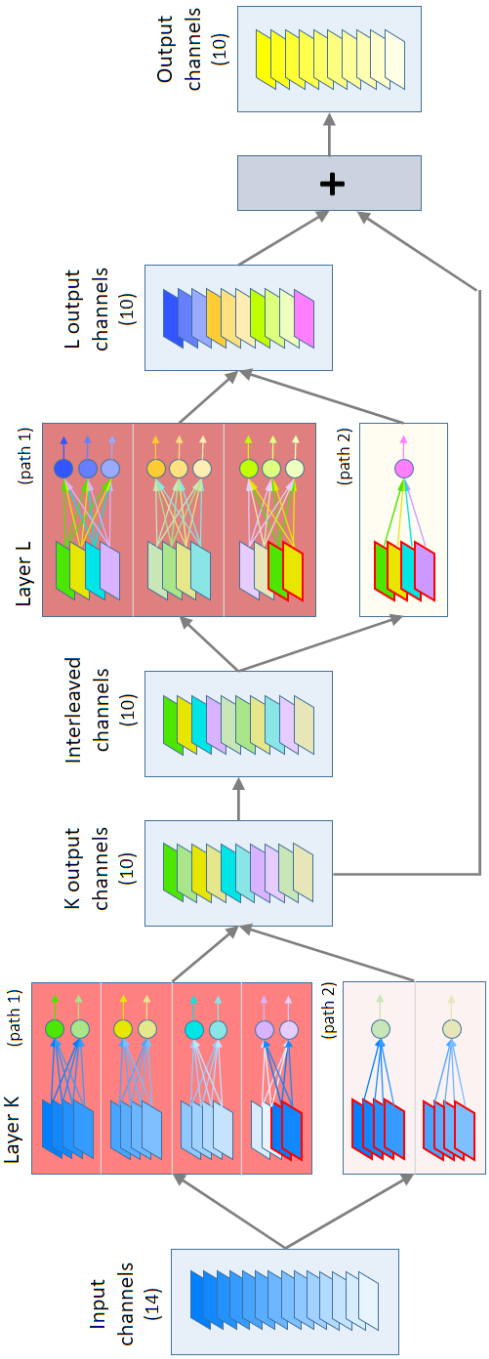

Figure 6.1: A schematic diagram of our pointwise convolution replacement. This example replaces a pointwise convolution with 14 input channels and 10 filters. It contains two convolutional layers, K and L, one interleaving and one summation layer. Channels surrounded by a red border represent replicated channels.

6.2 Methodology

6.2.1 Mathematical ground for regular pointwise convolutions

Let $X^i = \{x_1^i, x_2^i, \ldots, x_{Ic_i}^i\}$ be a set of input feature maps (2D lattices) for a convolutional layer i in a DCNN, where Ic_i denotes the number of input channels for this layer. Let $W^i = \{w_1^i, w_2^i, \ldots, w_{F_i}^i\}$ be a set of filters containing the weights for convolutions, where F_i denotes the number of filters at layer i, which is also the number of output channels of this layer. Following the notation proposed in Wang et al. (2019), a regular DCNN convolution can be mathematically expressed as in equation 6.1:

$$\begin{aligned} X^{i+1} &= W^i \bigotimes X^i \\ &= \{w_1^i * X^i,\ w_2^i * X^i, \ldots, w_{F_i}^i * X^i\} \end{aligned} \tag{6.1}$$

where the $\bigotimes$ operator indicates that filters in W^i are convolved with feature maps in X^i, using the $*$ operator to indicate a 3D tensor multiplication and shifting of a filter w_j^i across all patches of the size of the filter in all feature maps. For simplicity, we are ignoring the biased terms. Consequently, X^{i+1} will contain F_i feature maps that will feed the next layer $i+1$. The tensor shapes of involved elements are the following:

$$\begin{aligned} &X^i \in \mathcal{R}^{\mathcal{H}\times\mathcal{W}\times Ic_i} \\ &W^i \in \mathcal{R}^{F_i\times\mathcal{S}\times\mathcal{S}\times Ic_i} \ \rightarrow \ w_j^i \in \mathcal{R}^{\mathcal{S}\times\mathcal{S}\times Ic_i} \\ &X^{i+1} \in \mathcal{R}^{\mathcal{H}\times\mathcal{W}\times F_i} \end{aligned} \tag{6.2}$$

where $\mathcal{H} \times \mathcal{W}$ is the size (height, width) of feature maps, and $\mathcal{S} \times \mathcal{S}$ is the size of a filter (usually square). In this chapter, we work with $\mathcal{S} = 1$ because we focus on pointwise convolutions. In this case, each filter w_j^i carries Ic_i weights. The total number of weights P_i in layer i is obtained with a simple multiplication:

$$P_i = Ic_i \cdot F_i \tag{6.3}$$

6.2.2 Definition of grouped pointwise convolutions

For expressing a grouped pointwise convolution, let us split the input feature maps and the set of filters in G_i groups, as $X^i = \left\{X_1^i, X_2^i, \ldots, X_{G_i}^i\right\}$ and $W^i = \left\{W_1^i, W_2^i, \ldots, W_{G_i}^i\right\}$. Assuming that both Ic_i and F_i are divisible by G_i, the elements in X^i and W^i can be evenly distributed through all their subset X_j^i and W_j^i. Then, equation 6.1 can be reformulated as equation 6.4:

$$X^{i+1} \;=\; \left\{W_1^i \otimes X_1^i, W_2^i \otimes X_2^i, \ldots, W_{G_i}^i \otimes X_{G_i}^i\right\} \tag{6.4}$$

The shapes of the subsets are the following:

$$\begin{aligned} &X_m^i \in \mathcal{R}^{\mathcal{H}\times\mathcal{W}\times\frac{Ic_i}{G_i}} \\ &W_m^i \in \mathcal{R}^{Fg_i\times 1\times 1\times\frac{Ic_i}{G_i}} \;\rightarrow\; w_j^{i,m} \in \mathcal{R}^{1\times 1\times\frac{Ic_i}{G_i}} \end{aligned} \tag{6.5}$$

where Fg_i is the number of filters per group, namely, $Fg_i = {}^{F_i}/_{G_i}$. Since each filter $w_j^{i,m}$ only convolves on a fraction of input channels $({}^{Ic_i}/_{G_i})$, the total number of weights per subset W_m^i is $({}^{F_i}/_{G_i}) \cdot ({}^{Ic_i}/_{G_i})$. Multiplying the last expression by the number of groups provides the total number of weights $\overline{P_i}$ in a grouped pointwise convolutional layer i:

$$\overline{P_i} \;=\; (Ic_i \cdot F_i)/G_i \tag{6.6}$$

Eq. 6.6 shows that the number of trainable parameters is inversely proportional to the number of groups. However, the grouping has the evident drawback that it prevents the filters from being connected with all input channels, which reduces the possible connections of input channels for learning new patterns. As it may lead to a lower learning capacity of the DCNN, one must be cautious with using such a grouping technique.

6.2.3 Improved scheme for reducing the complexity of pointwise convolutions

Two major limitations of our previous method were inherited from constraints found in most deep learning APIs:

- The number of input channels Ic_i must be multiple of the number of groups G_i.
- The number of filters F_i must be multiple of the number of groups G_i.

The present work circumvents the first limitation by replicating channels from the input. The second limitation is circumvented by adding a second parallel path with another pointwise grouped convolution when required. Figure 6.1 shows an example of our updated architecture. Details of this process are described below, which is applied to substitute each pointwise convolutional layer i found in the original architecture. To explain the method, we start detailing the construction of layer K shown in Figure 6.1. For simplicity, we drop the index i and use the index K to refer to the original hyperparameters, i.e., we use Ic_K instead of Ic_i, F_K instead of F_i. Also, we will use the indexes $K1$ and $K2$ to refer to the parameters of the two parallel paths that may exist in layer K.

First of all, we must manually specify the value of the hyperparameter Ch. In the graphical example shown in Figure 6.1, we set $Ch = 4$. The rest of the hyperparameters, such as the number of groups in layers K and L, are determined automatically by the rules of our algorithm, according to the chosen value of Ch, the number of input channels Ic_K and the number of filters F_K. We do not have a procedure to find the optimal value of Ch. Hence we must apply ablation studies on a range of Ch values, as shown in the results section. For the example in Figure 6.1, we have chosen the value of Ch to obtain a full variety of situations that must be tackled by our algorithm, i.e., non-divisibility conditions.

6.2.4 Definition of layer K

The first step of the algorithm is to compute the number of groups in branch K1, as in equation 6.7:

$$G_{K1} = \left\lceil \frac{Ic_K}{Ch} \right\rceil \tag{6.7}$$

Since the number of input channels Ic_K may not be divisible by Ch, we use the ceiling operator on the division to obtain an integer number of groups. In the example, $G_{K1} = \lceil 14/4 \rceil = 4$. Thus, the output of filters in branch K1 can be defined as in 6.8:

$$K1 = \left\{ W_1^{K1} \otimes X_1^K, W_2^{K1} \otimes X_2^K, \ldots, W_{G_{K1}}^{K1} \otimes X_{G_{K1}}^K \right\} \tag{6.8}$$

The subsets X_m^K are composed of input feature maps x_j, collected in a sorted manner, i.e., $X_1^K = \{x_1,\ x_2, \ldots, x_{Ch}\}$, $X_2^K = \{x_{Ch+1},\ x_{Ch+2}, \ldots, x_{2Ch}\}$, etc. Equation 6.9 provides a general definition of which feature maps x_j are included in any feature subset X_m^K:

$$X_m^K = \{x_{a+1},\ x_{a+2}, \ldots, x_{a+Ch}\},\ a = (m-1) \cdot\ Ch \tag{6.9}$$

However, if Ic_K is not divisible by Ch, the last group $m = G_{K1}$ would not have Ch channels. In this case, the method will complete this last group by replicating $Ch - b$ initial input channels, where b is computed as stated in equation 6.10:

$$\begin{aligned} X_{G_{K1}}^K &= \{x_{a+1},\ x_{a+2}, \ldots, x_{a+b}, x_1, x_2, \ldots,\ x_{Ch-b}\}, \\ a &= (G_{K1} - 1) \cdot\ Ch, \\ b &= G_{K1} \cdot\ Ch - Ic_K \end{aligned} \tag{6.10}$$

It can be proved that b will always be less or equal than Ch, since b is the excess of the integer division Ic_K/Ch, i.e., $G_{K1} \cdot\ Ch$ will always be above or equal to Ic_K, but less than $Ic_K + Ch$, because otherwise G_{K1} would increase its value (as a quotient of Ic_K/Ch). In the example, $b = 2$, hence $X_4^{K1} = \{x_{13}, x_{14}, x_1, x_2\}$.

Then, the method calculates the number of filters per group Fg_{K1} as in 6.11:

$$Fg_{K1} = \left\lfloor \frac{F_K}{G_{K1}} \right\rfloor \tag{6.11}$$

To avoid divisibility conflicts, we have chosen the floor integer division this time. For the first path K1, each of the filter subsets shown in 6.8 will contain the following filters:

$$\begin{gathered} W_m^{K1} = \left\{ w_1^{K1,m},\ w_2^{K1,m}, \ldots, w_{Fg_{K1}}^{K1,m} \right\} \\ w_j^{K1,m} \in \mathcal{R}^{1\times 1\times Ch} \end{gathered} \tag{6.12}$$

For the first path of the example, the number of filters per group is $Fg_{K1} = \lfloor 10/4 \rfloor = 2$. So, the first path has 4 groups (G_{K1}) of 2 filters (Fg_{K1}), each filter being connected to 4 input channels (Ch).

If F_K is not divisible by Ch, a second path K2 will provide as many groups as filters not provided in K1, with one filter per group, to complete the total number of filters F_K:

$$\begin{aligned} G_{K2} &= F_K - Fg_{K1} \cdot\ G_{K1} \\ Fg_{K2} &= 1 \end{aligned} \tag{6.13}$$

In the example, $G_{K2} = 2$. The required input channels for the second path is $Ch \cdot G_{K2}$. The method obtains those channels reusing the same subsets of input feature maps X_m^K shown in 6.9. Hence, the output of filters in path $K2$ can be defined as in 6.14:

$$K2 = \left\{ w_1^{K2} * X_1^K, w_2^{K2} * X_2^K, \ldots, w_{G_{K2}}^{K2} * X_{G_{K2}}^K \right\} \tag{6.14}$$

where $w_j^{K2} \in \mathcal{R}^{1\times 1\times Ch}$. Therefore, each filter in $K2$ operates on the same subset of input channels as the corresponding subset of filters in $K1$. Hence, each filter in the second path can be considered as belonging to one of the groups of the first path.

It must be noticed that G_{K2} will always be less than G_{K1}. This is true because G_{K2} is the reminder of the integer division F_K/G_{K1}, as can be deduced from 6.11

and 6.13. This property warrants that there will be enough subsets X_m^K for this second path.

After defining paths K1 and K2 in layer K, the output of this layer is the concatenation of both paths:

$$K = \{K1,\ K2\} \tag{6.15}$$

The total number of channels after the concatenation is equal to $F_K = G_{K1} \cdot Fg_{K1} + G_{K2}$.

6.2.5 Interleaving stage

As mentioned above, grouped convolutions inherently face a limitation: each parallel group of filters computes its output from its subset of input channels, preventing combinations of channels connected to different groups. We propose to interleave the output channels from the convolutional layer K to alleviate this limitation.

The interleaving process simply consists in arranging the odd channels first and the even channels last, as noted in equation (16):

$$\begin{aligned} I^K =& \{k_1, k_3, k_5, \ldots, k_{2c-1}, \\ & k_2, k_4, k_6, \ldots, k_{2c}\} \\ c =& \lfloor F_K/2 \rfloor \end{aligned} \tag{6.16}$$

Here we are assuming that F_K is even. Otherwise, the list of odd channels will include an extra channel k_{2c+1}.

6.2.6 Definition of layer L

The interleaved output feeds the grouped convolutions in layer L to process data from more than one group from the preceding layer K.

To create layer L, we apply the same algorithm for layer K, but now the number of input channels is equal to F_K instead of Ic_K.

The number of groups in path L1 is computed as:

$$G_{L1} = \left\lceil \frac{F_K}{Ch} \right\rceil \tag{6.17}$$

Note that G_{L1} may not be equal to G_{K1}. In the example, $G_{L1} = \lceil 10/4 \rceil = 3$.

Then, the output of L1 is computed as in 6.18, where the input channel groups I_m^K come from the interleaving stage. Each group is composed of Ch channels, whose indexes are generically defined in 6.19:

$$L1 = \left\{ W_1^{L1} \otimes I_1^K, W_2^{L1} \otimes I_2^K, \ldots, W_{G_{L1}}^{K1} \otimes I_{G_{L1}}^K \right\} \tag{6.18}$$

$$\begin{aligned} I_m^K &= \left\{ i_{a+1}^K,\ i_{a+2}^K, \ldots, i_{a+Ch}^K \right\}, \\ a &= (m-1) \cdot\ Ch \end{aligned} \tag{6.19}$$

Again, the last group of indexes may not contain Ch channels due to a non-exact division condition in 6.17. Similar to path K1, for path L1, the missing channels in the last group will be supplied by replicating $Ch - b$ initial interleaved channels, where b is computed as stated in equation 6.20:

$$\begin{aligned} I_{G_{L1}}^K &= \left\{ i_{a+1}^K,\ i_{a+2}^K, \ldots, i_{a+b}^K, i_1^K, i_2^K, \ldots,\ i_{Ch-b}^K \right\}, \\ a &= (G_{L1} - 1) \cdot\ Ch, \\ b &= G_{L1} \cdot\ Ch\ - F_K \end{aligned} \tag{6.20}$$

The number of filters per group Fg_{L1} is computed as in 6.21:

$$Fg_{L1} = \left\lfloor \frac{F_K}{G_{L1}} \right\rfloor \tag{6.21}$$

In the example, $Fg_{L1} = \lfloor 10/3 \rfloor = 3$. Each group of filters W_m^{L1} shown in 6.18 can be defined as in 6.22, each one containing Fg_{L1} convolutional filters of Ch inputs:

$$\begin{aligned} W_m^{L1} &= \left\{ w_1^{L1,m},\ w_2^{L1,m}, \ldots, w_{Fg_{L1}}^{L1,m} \right\} \\ w_j^{L1,m} &\in \mathcal{R}^{1 \times 1 \times Ch} \end{aligned} \tag{6.22}$$

It should be noted that if the division in 6.21 is not exact, the number of output channels from layer L may not reach the required F_K outputs. In this case, a second

path L2 will be added, with the following parameters:

$$\begin{aligned} G_{L2} &= F_K - Fg_{L1} \cdot G_{L1} \\ Fg_{L2} &= 1 \end{aligned} \tag{6.23}$$

In the example, $G_{L2} = 1$. The output of path L2 is computed as in 6.24, defining one extra convolutional filter for some initial groups of interleaved channels declared in 6.18 and 6.19, taking into account that G_{L2} will always be less than G_{L1} according to the same reasoning done for G_{K2} and G_{K1}:

$$L2 = \left\{ w_1^{L2} * I_1^K, w_2^{L2} * I_2^K, \ldots, w_{G_{L2}}^{L2} * I_{G_{L2}}^K \right\} \tag{6.24}$$

The last step in defining the output of layer L is to join the outputs of paths L1 and L2:

$$L = \{L1,\ L2\} \tag{6.25}$$

6.2.7 Joining of layers

Finally, the output of both convolutional layers K and L are summed to create the output of the original layer:

$$X^{i+1} = K + L \tag{6.26}$$

Compared to concatenation, summation has the advantage of allowing residual learning in the filters of layer L because gradient can be backpropagated through L filters or directly to K filters. In other words, residual layers provide more learning capacity with a low degree of downsides due to increasing the number of layers (i.e., overfitting, longer training time, etc.) Table 6.5 shows an ablation study that contains experiments done without the interleaving and the L layers (rows labeled with "no L"). These experiments empirically prove that the interleaving mechanism and the secondary L layer help improve the low-impact sub-architecture accuracy.

It is worth mentioning that we only add layer L and the interleaving when the

number of input channels is bigger or equal to the number of filters in layer K.

6.2.8 Computing the number of parameters

We can compute the total number of parameters of our sub-architecture. First, equation 6.27 shows that the number of filters in layer K is equal to the number of filters in layer L, which in turn is equal to the total number of filters in the original convolutional layer F_i:

$$Fg_{K1} \cdot \ G_{K1} + G_{K2} = \ Fg_{L1} \cdot \ G_{L1} + G_{L2} = F_i \tag{6.27}$$

Then, the total number of parameters $\overline{\overline{P_i}}$ is twice the number of original filters multiplied by the number of input channels per filter:

$$\overline{\overline{P_i}} \ = \ 2(F_i \cdot Ch) \tag{6.28}$$

Therefore, comparing equation 6.28 with 6.3, it is clear that Ch must be significantly less than $Ic_i/2$ to reduce the number of parameters of a regular pointwise convolutional layer. Also, comparing equation 6.28 with 6.6, our sub-architecture provides a parameter reduction similar to a plain grouped convolutional layer when Ch is around $Ic_i/2G_i$. However, we cannot specify a general G_i term because of the complexity of our pair of layers with possibly two paths per layer.

The requirement for a low value of Ch is also necessary to ensure that divisions in equations 6.7 and 6.17 provide quotients above one. Otherwise, our method will not create a grouping. Hence, Ch must be less or equal to $Ic_i/2$ and $F_i/2$. These are the only two constraints that our method is restricted by.

As shown in Table 6.1, pointwise convolutional layers found in real networks such as EfficientNet-B0 have significant Figures for Ic_i and F_i, either hundreds or thousands. Therefore, values of Ch less or equal to 32 will ensure a good ratio of parameter reduction for most of these pointwise convolutional layers.

EfficientNet is one of the most complex (but efficient) architectures found in the literature. To our method, the degree of complexity of a DCNN is mainly related

to the maximum number of input channels and output features in any pointwise convolutional layer. Our method does not care about the number of layers, in-depth or parallel, because it works on each layer independently. Therefore, the complexity of EfficientNet-B0 can be considered significantly high, considering the values shown in the last row of Table 6.1. Arguably, other versions of EfficientNet (B1, B2, etc.) and other types of DCNN can exceed those values. In such cases, higher values of Ch may be necessary, but we cannot provide any rule to forecast its optimum value for any pointwise convolutional layer configuration.

Table 6.1: For a standard pointwise convolution with Ic input channels, F filters, P parameters, and a given number of channels per group Ch, this Table shows the calculated parameters for layers K and L: the number of groups $G_{<layer><path>}$ and the number of filters per group $Fg_{<layer><path>}$. The last 2 columns show the total number of parameters and their percentage concerning the original layer.

Original Settings				Layer K			Layer L			K+L Params	
Ic	F	P	Ch	G_{K1}	Fg_{K1}	G_{K2}	G_{L1}	Fg_{L1}	G_{L2}	Total	%
14	10	140	4	4	2	2	3	3	1	80	57.14%
160	3,840	614,400	16	10	384	0	0	0	0	61,440	10.00%
			32	5	768	0	0	0	0	122,880	20.00%
192	1,152	221,184	16	12	96	0	0	0	0	18,432	8.33%
			32	6	192	0	0	0	0	36,864	16.67%
1,152	320	368,640	16	72	4	32	20	16	0	10,240	2.78%
			32	36	8	32	10	32	0	20,480	5.56%
3,840	640	2,457,600	16	240	2	160	40	16	0	20,480	0.83%
			32	120	5	40	20	32	0	40,960	1.67%

6.2.9 Activation Function

In 2018, Prajit et. al. Ramachandran et al. (2017) tested many activation functions. In their experimentation, they found that the best performing one was the so-called "swish," shown in equation 6.29.

$$f(x) = x \cdot sigmoid(\beta x) \tag{6.29}$$

In previous chapters, we used the ReLU activation function. In this work, we use the swish activation function. This change gives us better results in our ablation

experiments shown in Table 6.5.

6.2.10 Implementation Details

We tested our optimization by replacing original pointwise convolutions in the EfficientNet-B0 and named it "kEffNet-B0 V2". With CIFAR-10, we tested an additional modification that skips the first 4 convolutional strides, allowing input images with 32x32 pixels instead of the original resolution of 224x224 pixels.

In all our experiments, we saved the trained network from the epoch that achieved the lowest validation loss for testing with the test dataset. Convolutional layers are initialized with Glorot's method Glorot and Bengio (2010). All experiments were trained with RMSProp optimizer, data augmentation Shorten and Khoshgoftaar (2019), and cyclical learning rate schedule Smith (2017). We worked with various configurations of hardware with NVIDIA video cards. Regarding software, we did our experiments with K-CAI Schuler (2021) and Keras Chollet et al. (2015) on the top of Tensorflow Abadi et al. (2015).

Our source code is publicly available at https://github.com/joaopauloschuler/kEffNetV2/ .

6.2.11 Horizontal Flip

In some experiments, we run the model twice with the input image and its horizontally flipped version. The output from the softmax from both runs is summed before class prediction. In these experiments, the number of floating-point computations doubles, although the number of trainable parameters remains the same.

6.3 Results and Discussion

In this section, we present and discuss the results of the proposed scheme with three image classification datasets: CIFAR-10 dataset Krizhevsky (2009), Malaria dataset Rajaraman et al. (2018), and colorectal cancer histology dataset Kather et al. (2016).

6.3.1 Results on the CIFAR-10 dataset

This CIFAR-10 dataset originally had 50k images for training and 10k for testing. We picked 5k images for validation and left the training set with 45k images. We run experiments with 50 and 180 epochs.

On Table 6.2 we compare kEffNet-B0 V1 (our previous method) and V2 (our current method), for two values of Ch. We can see that our V2 models have slightly more reduction in both numbers of parameters and floating-point computations than the V1 counterpart models while achieving slightly higher accuracy. Specifically, V2 models save 10% of the parameters (from 1,059,202 to 950,650) and 11% of the floating-point computations (from 138,410,206 to 123,209,110) of V1 models. All of our variants obtain similar accuracy to the baseline with a remarkable reduction of resources (at least 26.3% of trainable parameters and 35.5% of computations).

Table 6.2: Comparing EfficientNet-B0, kEffNet-B0 V1 and kEffNet-B0 V2 with CIFAR-10 dataset after 50 epochs.

Model	Parameters	%	Computations	%	Test acc.
EfficientNet-B0 baseline	4,020,358	100.0%	389,969,098	100.0%	93.33%
kEffNet-B0 V1 16ch	639,702	15.9%	84,833,890	21.8%	92.46%
kEffNet-B0 V2 16ch	623,226	15.5%	82,804,374	21.2%	92.61%
kEffNet-B0 V1 32ch	1,059,202	26.3%	138,410,206	35.5%	93.61%
kEffNet-B0 V2 32ch	950,650	23.6%	123,209,110	31.6%	**93.67%**

As the scope of this work is limited to small datasets and architectures, we only experimented with the smallest EfficientNet variant (EfficientNet-B0) and our modified variant (kEffNet-B0). Nevertheless, Table 6.3 provides the number of trainable parameters of the other EfficientNet variants (original and parameter-reduced). Equation 6.3 indicates that the number of parameters grows with the number of filters and input channels. Equation 6.6 indicates that the number of parameters decreases with the number of groups. As we create more groups when the number of input channels grows, we expect to find bigger parameter savings on larger models. This saving can be seen in Table 6.3.

We also tested our kEffNet-B0 with 2, 4, 8, 16, and 32 channels per group for 50 epochs, as shown in Table 6.4. As expected, the test classification accuracy increases when allocating more channels per group: from 84.26% for Ch=2 to 93.67% for

Table 6.3: Number of trainable parameters for EfficientNet, kEffNet V2 16ch, and kEffNet V2 32ch with a 10 classes dataset.

variant	EfficientNet	kEffNet V2 16ch	%	kEffNet V2 32ch	%
B0	4,020,358	623,226	15.50%	950,650	23.65%
B1	6,525,994	968,710	14.84%	1,389,062	21.29%
B2	7,715,084	983,198	12.74%	1,524,590	19.76%
B3	10,711,602	1,280,612	11.96%	2,001,430	18.68%
B4	17,566,546	1,858,440	10.58%	2,911,052	16.57%
B5	28,361,274	2,538,870	8.95%	4,011,626	14.14%
B6	40,758,754	3,324,654	8.16%	5,245,140	12.87%
B7	63,812,570	4,585,154	7.19%	7,254,626	11.37%

Ch=32. Also, the resource-saving decreases as the number of channels per group increase: from 7.8% of parameters and 11.4% of computations for Ch=2 to 23.6% of parameters and 31.6% of computations for Ch=32 (compared to the baseline). For CIFAR-10, if we aim to achieve an accuracy comparable to the baseline, we must choose at least 16 channels per group. If we add an extra run per image sample with horizontal flipping when training kEffNet-B0 V2 32ch, the classification accuracy increases from 93.67% to 94.01%.

Table 6.4: Ablation study done with the CIFAR-10 dataset for 50 epochs, comparing the effect of varying the number of channels per group. It also includes the improvement achieved by double training kEffNet-B0 V2 32ch with original and horizontally flipped images.

Model	Parameters	%	Computations	%	Test acc.
EfficientNet-B0 baseline	4,020,358	100.0%	389,969,098	100.0%	93.33%
kEffNet-B0 V2 2ch	311,994	7.8%	44,523,286	11.4%	84.36%
kEffNet-B0 V2 4ch	354,818	8.8%	49,487,886	12.7%	87.66%
kEffNet-B0 V2 8ch	444,346	11.1%	60,313,526	15.5%	90.53%
kEffNet-B0 V2 16ch	623,226	15.5%	82,804,374	21.2%	92.61%
kEffNet-B0 V2 32ch	950,650	23.6%	123,209,110	31.6%	93.67%
kEffNet-B0 V2 32ch + H Flip	950,650	23.6%	246,418,220	63.3%	**94.01%**

Table 6.5 replicates most of the results shown in Table 6.4 but compares the effect of not including layer L and interleaving substituting the swish activation function with the typical ReLU. As can be observed, disabling layer L has a noticeable degradation in test accuracy when the values of Ch are smaller. For example, when Ch=4, the performance drops more than 5%. On the other hand, when Ch=32, the drop is less than 0.5%. This is logical, considering that the more channels are included per group, the more chances are to combine input features in the filters.

Therefore, a second layer and the corresponding interleaving are not as crucial as when the filters of layer K are fed with fewer channels.

The same effect can be appreciated in comparing activation functions: the swish function works better than the ReLU function. Still, it provides less improvement for a larger number of channels per group. Nevertheless, the gain in the least difference case (32 ch) is still profitable, with more than 1.5% of extra test accuracy when using the swish activation function.

Table 6.5: Extra experiments made for kEffNet-B0 V2 4ch, 8ch, 16ch and 32ch variants. Rows labeled with "no L" indicate experiments using only layer K, i.e., disabling layer L and interleaving. Rows labeled with "ReLU" replace the swish activation function by ReLU.

Model	Parameters	%	Computations	%	Test acc.
EfficientNet-B0 baseline	4,020,358	100.0%	389,969,098	100.0%	93.33%
kEffNet-B0 V2 4ch	354,818	8.8%	49,487,886	12.7%	87.66%
kEffNet-B0 V2 4ch no L	342,070	8.5%	48,064,098	12.3%	82.44%
kEffNet-B0 V2 4ch ReLU	354,818	8.8%	47,595,914	12.2%	85.34%
kEffNet-B0 V2 8ch	444,346	11.1%	60,313,526	15.5%	90.53%
kEffNet-B0 V2 8ch no L	422,886	10.5%	57,466,370	14.7%	89.27%
kEffNet-B0 V2 8ch ReLU	444,346	11.1%	58,421,554	15.0%	88.82%
kEffNet-B0 V2 16ch	623,226	15.5%	82,804,374	21.2%	92.61%
kEffNet-B0 V2 16ch no L	584,934	14.6%	77,356,802	19.8%	91.52%
kEffNet-B0 V2 16ch ReLU	623,226	15.5%	80,912,406	20.8%	91.16%
kEffNet-B0 V2 32ch	950,650	23.6%	123,209,110	31.6%	**93.67**%
kEffNet-B0 V2 32ch no L	879,750	21.9%	112,684,706	28.9%	93.21%
kEffNet-B0 V2 32ch ReLU	950,650	23.7%	121,317,142	31.1%	92.00%

Table 6.6 shows the effect in accuracy when classifying the CIFAR-10 dataset with EfficientNet-B0 and our kEffNet-B0 V2 32ch variant for 180 epochs instead of 50 epochs. The additional training epochs assign slightly higher test accuracy to the baseline than our core variant. When adding horizontal flipping, our variant has slightly surpassed the baseline results. Nevertheless, all three results can be considered similar, but our variant offers a significant saving in parameters and computations. Although the H flipping doubles the computational cost of our core variant, it remains only a fraction (63.3%) of the baseline computational cost.

Table 6.6: Results obtained with the CIFAR-10 dataset after 180 epochs.

Model	Parameters	%	Computations	%	Test acc.
EfficientNet-B0 baseline	4,020,358	100.0%	389,969,098	100.0%	94.86%
kEffNet-B0 V2 32ch	950,650	23.6%	123,209,110	31.6%	94.45%
kEffNet-B0 V2 32ch + H Flip	950,650	23.6%	246,418,220	63.3%	**94.95%**

6.3.2 Results on the Malaria dataset

The Malaria dataset Rajaraman et al. (2018) has 27,558 cell images from infected and healthy cells separated into 2 classes. There is the same number of images for healthy and infected cells. From the original 27,558 images set, we separated 10% of the images (2756 images) for validation and another 10% for testing. On all training, validation, and test subsets, there are 50% of healthy cell images. We quadruplicated the number of validation images by flipping these images horizontally and vertically, resulting in 11,024 images for validation.

On this dataset, we tested our kEffNet-B0 with 2, 4, 8, 12, 16, and 32 channels per group and the baseline architecture, as shown in Table 6.7. Our variants have from 7.5% to 23.5% of the trainable parameters and from 15.7% to 42.2% of the computations allocated by the baseline architecture. Although the worst classification accuracy was found with the smallest variant (2ch), its classification accuracy is less than 1% inferior to the best performing variant (16ch) and only 0.69% below the baseline performance. With only 8 channels per group, our method equals the baseline accuracy with a small portion of the parameters (10.8%) and computations (22.5%) required by the baseline architecture. Curiously, our 32ch variant is slightly worse than the 16ch variant but better than the baseline. It is an example that a relatively low complexity of the input images may require fewer channels per filter (and more parallel groups of filters), to capture the relevant features of images optimally.

6.3.3 Results on the Colorectal cancer histology dataset

The collection of samples in the colorectal cancer histology dataset Kather et al. (2016) contains 5000 150x150 images separated into 8 classes: adipose, complex,

Table 6.7: Results obtained with the Malaria dataset after 75 epochs.

Model	Parameters	%	Computations	%	Test acc.
EfficientNet-B0 baseline	4,010,110	100.0%	389,958,834	100.0%	97.39%
kEffNet-B0 V2 2ch	301,746	7.5%	61,196,070	15.7%	96.70%
kEffNet-B0 V2 4ch	344,570	8.6%	69,691,358	17.9%	96.95%
kEffNet-B0 V2 8ch	434,098	10.8%	87,725,254	22.5%	97.39%
kEffNet-B0 V2 12ch	524,026	13.1%	106,199,566	27.2%	97.31%
kEffNet-B0 V2 16ch	612,978	15.3%	124,672,934	32.0%	**97.61%**
kEffNet-B0 V2 32ch	940,402	23.5%	164,422,950	42.2%	97.57%

debris, empty, lympho, mucosa, stroma, and tumor. Similar to what we did with the Malaria dataset, we separated 10% of the images for validation and another 10% for testing. We also quadruplicated the number of validation images by flipping these images horizontally and vertically.

On this dataset, we tested our kEffNet-B0 with 2, 4, 8, 12, and 16 channels per group and the baseline architecture, as shown in Table 6.8. Similar to the Malaria dataset, higher values of channels per group do not lead to better performance. In this case, the variants with the highest classification accuracy are 4ch and 8ch, achieving 98.02% of classification accuracy, outperforming the baseline accuracy in 0.41%. The 16ch variant has obtained the same accuracy as the 2ch variant but doubles the required resources. Again, it indicates that the complexity of the images plays a role in selecting the optimal number of channels per group. In other words, simpler images may require fewer channels per group. Unfortunately, the only method we know to find this optimal value is performing these scanning experiments.

Table 6.8: Results obtained with the colorectal cancer dataset after 1000 epochs.

Model	Parameters	%	Computations	%	Test acc.
EfficientNet-B0 baseline	4,017,796	100.0%	389,966,532	100.0%	97.61%
kEffNet-B0 V2 2ch	355,064	8.8%	61,203,768	15.7%	97.62%
kEffNet-B0 V2 4ch	397,888	9.9%	69,699,056	17.9%	**98.02%**
kEffNet-B0 V2 8ch	487,416	12.1%	87,732,952	22.5%	**98.02%**
kEffNet-B0 V2 12ch	531,712	13.2%	106,207,264	27.2%	97.22%
kEffNet-B0 V2 16ch	620,664	15.4%	124,680,632	32.0%	97.62%

6.4 Conclusion

This chapter presents an efficient scheme for decreasing the complexity of pointwise convolutions in DCNNs for image classification based on interleaved grouped filters with no divisibility constraints. From our experiments, we can conclude that connecting all input channels from the previous layer to all filters is unnecessary: grouped convolutional filters can achieve the same learning power with a small fraction of resources (1/3 of floating-point computations, 1/4 of parameters). Our enhanced scheme avoids the divisibility constraints, further reducing the required resources (up to 10% less) while maintaining or slightly surpassing the accuracy of our previous method.

We have made ablation studies to obtain the optimal number of channels per group for each dataset. For the colorectal cancer dataset, this number is surprisingly low (4 channels per group). Conversely, CIFAR-10's best results require at least 16 channels per group. This fact indicates that the input images' complexity affects our sub-architectures optimal configuration.

CHAPTER 7

Concluding remarks

Not all input channels need to be connected in every convolutional filter. Parallel groups of filters can learn all the required features for image classification. We tested this idea in the first layers of a DCNN via separating achromatic and chromatic branches and pointwise grouping filters.

7.1 Thesis highlights

In chapter 2, by converting RGB images to CIE Lab images and then loading chromatic and achromatic values into their separated branches, one for L and another for AB, we show that the first layer of any CNN can be optimized by reducing more than 50% the original number of weights while keeping or even improving the classification accuracy.

Instead of separating only the first layer, in chapter 3, the first three convolutional

layers of a modified Inception V3 were split into chromatic and achromatic branches. This architecture reduces learnable parameters and floating point operations from 1/3 to 1/2 in these initial three layers. This architecture achieves state-of-the-art classification accuracy with the PlantVillage dataset. For this dataset, our experiments indicate that from 50% to 80% of the filters should enter the chromatic branch indicating that color is essential for this task.

In chapter 4, we showed that our two-branch CNN for plant disease classification proposed in chapter 3 is more reliable than a typical CNN based on RGB when input images suffer from noise. Our 20%L-80%AB and 50%L-50%AB models better classify input images with noise by margins up to 10%.

In chapter 5, we introduced a parameter and computationally efficient replacement for pointwise convolutions. For example, our optimized EfficientNet-B0, named kEffNet-B0 32ch, achieves a slightly higher classification accuracy while saving 74% of the trainable parameters and 55% of the floating-point computations. Our pointwise optimization works better on architectures containing depthwise and pointwise convolutions.

In chapter 6, we further reduced the required floating-point computations and trainable parameters (up to 10% less) while keeping or slightly improving the accuracy found in chapter 5. We did ablation studies to obtain the optimal number of channels per group for each tested dataset. We found that the input images' complexity affects our optimization's optimal configuration.

From chapter 2 to chapter 6 of this thesis, our experiments show that not all input channels need to be connected in every convolutional filter. In image classification, parallel filters can learn the relevant features while significantly reducing the number of parameters and floating-point computations.

7.2 Future research lines

Chapters 2, 3 and 4 shown an optimization that separates achromatic and chromatic information up to 3 layers deep. How many layers deep the two-branch approach

could go remains as an open and interesting question.

Chapter 4 shows that the two-branch approach provides better noise resistance with the PlantVillage dataset. Although this approach would likely work with other datasets and baseline architectures, these experiments are yet to be made.

Chapters 5 and 6 show an optimization for pointwise convolutions. A similar architecture could possibly be used for optimizing spatial convolutions on existing architectures.

Also, in chapters 5 and 6, we currently cannot determine the optimal number of channels per group Ch automatically, according to the complexity of each pointwise convolutional layer and the training dataset. At this moment, Ch is a hyperparameter that requires tuning for each application. Finding an automatic method to discover Ch would make the usage of our optimization easier.

The scope of this thesis is limited to image classification. Considering that the optimizations shown in this thesis would work in other problem domains such as semantic image segmentation and image generation is plausible. Other fields not related to image processing that apply CNNs, such as natural language processing, might also benefit from approaches similar to the approaches detailed in this thesis.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. URL: https://www.tensorflow.org/. software available from tensorflow.org.

Amara, J., Bouaziz, B., Algergawy, A., 2017. A deep learning-based approach for banana leaf diseases classification, in: Datenbanksysteme für Business, Technologie und Web – Workshopband, pp. 79–88.

Baykal, C., Liebenwein, L., Gilitschenski, I., Feldman, D., Rus, D., 2019. Data-dependent coresets for compressing neural networks with applications to generalization bounds, in: International Conference on Learning Representations. URL: https://openreview.net/forum?id=HJfwJ2A5KX.

Bianco, S., Cusano, C., Napoletano, P., Schettini, R., 2017. Improving cnn-based

texture classification by color balancing. Journal of Imaging 3. URL: https://www.mdpi.com/2313-433X/3/3/33, doi:10.3390/jimaging3030033.

Chaudhary, P., Chaudhari, A., Godara, S., 2012. Color transform based approach for disease spot detection on plant leaf. International Journal of Computer Science and Telecommunications 3, 65–70.

Chen, Y., Li, J., Xiao, H., Jin, X., Yan, S., Feng, J., 2017. Dual path networks 30. URL: https://proceedings.neurips.cc/paper/2017/file/f7e0b956540676a129760a3eae309294-Paper.pdf.

Cheng, G., Guo, W., 2017. Rock images classification by using deep convolution neural network. Journal of Physics: Conference Series 887, 012089. URL: https://doi.org/10.1088/1742-6596/887/1/012089, doi:10.1088/1742-6596/887/1/012089.

Cheng, Y., Yu, F.X., Feris, R.S., Kumar, S., Choudhary, A.N., Chang, S., 2015. An exploration of parameter redundancy in deep networks with circulant projections, in: 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015, IEEE Computer Society. pp. 2857–2865. URL: https://doi.org/10.1109/ICCV.2015.327, doi:10.1109/ICCV.2015.327.

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

Denil, M., Shakibi, B., Dinh, L., Ranzato, M., de Freitas, N., 2013. Predicting parameters in deep learning, in: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, Curran Associates Inc., Red Hook, NY, USA. p. 2148–2156.

Ferentinos, K.P., 2018. Deep learning models for plant disease detection and diagnosis. Computers and Electronics in Agriculture 145, 311 – 318. URL: http://www.sciencedirect.com/science/article/pii/S0168169917311742, doi:https://doi.org/10.1016/j.compag.2018.01.009.

Fuentes, A., Yoon, S., Kim, S.C., Park, D.S., 2017. A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition. Sensors 17. URL: https://www.mdpi.com/1424-8220/17/9/2022, doi:10.3390/s17092022.

Fujita, E., Kawasaki, Y., Uga, H., Kagiwada, S., Iyatomi, H., 2016. Basic investigation on a robust and practical plant diagnostic system, in: 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 989–992. doi:10.1109/ICMLA.2016.0178.

Fukushima, K., 1980. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics 36, 193–202. URL: https://doi.org/10.1007/BF00344251, doi:10.1007/BF00344251.

G., G., J., A.P., 2019. Identification of plant leaf diseases using a nine-layer deep convolutional neural network. Computers & Electrical Engineering 76, 323 – 338. URL: http://www.sciencedirect.com/science/article/pii/S0045790619300023, doi:https://doi.org/10.1016/j.compeleceng.2019.04.011.

Gevers, T., Gijsenij, A., van de Weijer, J., Geusebroek, J.M., 2012. Color in Computer Vision: Fundamentals and Applications. 1st ed., Wiley Publishing. doi:10.5555/2432392.

Glorot, X., Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks 9, 249–256. URL: https://proceedings.mlr.press/v9/glorot10a.html.

Gowda, S.N., Yuan, C., 2019. Colornet: Investigating the importance of color spaces for image classification, in: Jawahar, C., Li, H., Mori, G., Schindler, K. (Eds.), Computer Vision – ACCV 2018, Springer International Publishing, Cham. pp. 581–596.

Han, S., Mao, H., Dally, W.J., 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. International Conference on Learning Representations (ICLR) .

He, K., Zhang, X., Ren, S., Sun, J., 2016a. Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Los Alamitos, CA, USA. pp. 770–778. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.90, doi:10.1109/CVPR.2016.90.

He, K., Zhang, X., Ren, S., Sun, J., 2016b. Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778. doi:10.1109/CVPR.2016.90.

Hering, E., 1920. Outlines of a Theory of the Light Sense. Cambridge: Harvard University Press.

Howard, A., Sandler, M., Chen, B., Wang, W., Chen, L.C., Tan, M., Chu, G., Vasudevan, V., Zhu, Y., Pang, R., Adam, H., Le, Q., 2019. Searching for mobilenetv3, in: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 1314–1324. doi:10.1109/ICCV.2019.00140.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. CoRR abs/1704.04861. URL: http://arxiv.org/abs/1704.04861, arXiv:1704.04861.

Huang, G., Liu, Z., Weinberger, K.Q., 2017. Densely connected convolutional networks, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE. pp. 2261–2269. doi:10.1109/CVPR.2017.243.

Hughes, D.P., Salath'e , M., 2015. An open access repository of images on plant health to enable the development of mobile disease diagnostics through machine learning and crowdsourcing. CoRR abs/1511.08060. URL: http://arxiv.org/abs/1511.08060, arXiv:1511.08060.

Ioannou, Y., Robertson, D.P., Cipolla, R., Criminisi, A., 2017. Deep roots: Improving cnn efficiency with hierarchical filter groups , 5977–5986URL: https://

doi.ieeecomputersociety.org/10.1109/CVPR.2017.633, doi:10.1109/CVPR.2017.633.

Johannes, A., Picon, A., Alvarez-Gila, A., Echazarra, J., Rodriguez-Vaamonde, S., Navajas, A.D., Ortiz-Barredo, A., 2017. Automatic plant disease diagnosis using mobile capture devices, applied on a wheat use case. Computers and Electronics in Agriculture 138, 200–209. URL: https://www.sciencedirect.com/science/article/pii/S016816991631050X, doi:https://doi.org/10.1016/j.compag.2017.04.013.

Kahatapitiya, K., Rodrigo, R., 2021. Exploiting the redundancy in convolutional filters for parameter reduction, in: 2021 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1409–1419. doi:10.1109/WACV48630.2021.00145.

Kather, J.N., Zöllner, F.G., Bianconi, F., Melchers, S.M., Schad, L.R., Gaiser, T., Marx, A., Weis, C.A., 2016. Collection of textures in colorectal cancer histology. URL: https://doi.org/10.5281/zenodo.53169, doi:10.5281/zenodo.53169.

Krizhevsky, A., 2009. Learning multiple layers of features from tiny images. Technical Report. University of Toronto. Toronto, Ontario.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks, in: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (Eds.), Advances in Neural Information Processing Systems 25. Curran Associates, Inc., pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D., 1989a. Backpropagation applied to handwritten zip code recognition. Neural Computation 1, 541–551.

LeCun, Y., Denker, J., Solla, S., 1989b. Optimal brain damage, in: Touretzky, D. (Ed.), Advances in Neural Information Processing Systems,

Morgan-Kaufmann. URL: https://proceedings.neurips.cc/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf.

Liebenwein, L., Baykal, C., Carter, B., Gifford, D., Rus, D., 2021. Lost in pruning: The effects of pruning neural networks beyond test accuracy. Proceedings of Machine Learning and Systems 3.

Liebenwein, L., Baykal, C., Lang, H., Feldman, D., Rus, D., 2020. Provable filter pruning for efficient neural networks, in: International Conference on Learning Representations. URL: https://openreview.net/forum?id=BJxkOlSYDH.

Lin, M., Chen, Q., Yan, S., 2014. Network in network. arXiv:1312.4400.

Maeda-Gutiérrez, V., Galván Tejada, C., Zanella Calzada, L., Celaya Padilla, J., Galván Tejada, J., Gamboa-Rosales, H., Luna-Garcia, H., Magallanes-Quintanar, R., Carlos, G.M., Olvera-Olvera, C., 2020. Comparison of convolutional neural network architectures for classification of tomato plant diseases. Applied Sciences 10. URL: https://www.mdpi.com/2076-3417/10/4/1245, doi:10.3390/app10041245.

Mody, M., Mathew, M., Jagannathan, S., 2017. Efficient pre-processor for cnn. Proc. IS&T Int'l. Symp. on Electronic Imaging: Autonomous Vehicles and Machines , 50–53URL: https://doi.org/10.2352/ISSN.2470-1173.2017.19.AVM-020, doi:10.2352/ISSN.2470-1173.2017.19.AVM-020.

Mohanty, S.P., Hughes, D.P., Salathé, M., 2016. Using deep learning for image-based plant disease detection. Frontiers in Plant Science 7, 1419. URL: https://www.frontiersin.org/article/10.3389/fpls.2016.01419, doi:10.3389/fpls.2016.01419.

Nakatani, H., 1942. Segmentation of color aerial photographs using hsv color models, in: Proc. IAPR Work. Mach. Vis. Appl. MVA, p. 367–370. URL: http://b2.cvl.iis.u-tokyo.ac.jp/mva/proceedings/CommemorativeDVD/1992/papers/1992367.pdf.

Napoletano, P., 2017. Hand-crafted vs learned descriptors for color texture classification, in: Bianco, S., Schettini, R., Trémeau, A., Tominaga, S. (Eds.),

Computational Color Imaging, Springer International Publishing, Cham. pp. 259–271. doi:10.1007/978-3-319-56010-6_22.

Ngugi, L.C., Abelwahab, M., Abo-Zahhad, M., 2021. Recent advances in image processing techniques for automated leaf pest and disease recognition – a review. Information Processing in Agriculture 8, 27–51. URL: https://www.sciencedirect.com/science/article/pii/S2214317320300196, doi:https://doi.org/10.1016/j.inpa.2020.04.004.

Ohta, Y.I., Kanade, T., Sakai, T., 1980. Color information for region segmentation. Computer Graphics and Image Processing 13, 222–241. URL: https://www.sciencedirect.com/science/article/pii/0146664X80900477, doi:https://doi.org/10.1016/0146-664X(80)90047-7.

Parkhi, O.M., Vedaldi, A., Zisserman, A., Jawahar, C.V., 2012. Cats and dogs, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3498–3505. doi:10.1109/CVPR.2012.6248092.

Pouli, T., Reinhard, E., Cunningham, D.W., 2013. Image Statistics in Visual Computing. 1st ed., A. K. Peters, Ltd., USA.

Rajaraman, S., Antani, S., Poostchi, M., Silamut, K., Hossain, M., Maude, R., Jaeger, S., Thoma, G., 2018. Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images. PeerJ 6. doi:10.7717/peerj.4568.

Ramachandran, P., Zoph, B., Le, Q.V., 2017. Searching for activation functions. CoRR abs/1710.05941. URL: http://arxiv.org/abs/1710.05941, arXiv:1710.05941.

Ramcharan, A., Baranowski, K., McCloskey, P., Ahmed, B., Legg, J., Hughes, D.P., 2017. Deep learning for image-based cassava disease detection. Frontiers in Plant Science 8, 1852. URL: https://www.frontiersin.org/article/10.3389/fpls.2017.01852, doi:10.3389/fpls.2017.01852.

Reed, R., 1993. Pruning algorithms-a survey. IEEE Transactions on Neural Networks 4, 740–747. doi:10.1109/72.248452.

Rijsbergen, C.J.V., 1979. Information Retrieval. 2nd ed., Butterworth-Heinemann, USA.

Robertson, A.R., 1992. Color perception. Physics Today 45, 24–29. URL: https://doi.org/10.1063/1.881324, doi:10.1063/1.881324, arXiv:https://doi.org/10.1063/1.881324.

Romaní, S., 2006. Labeled Color Image Segmentation through Perceptually Relevant Chromatic Patterns. Ph.D. thesis. doi:10.13140/2.1.2614.6882.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L., 2015. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) 115, 211–252. doi:10.1007/s11263-015-0816-y.

Schuler, J., Romaní, S., Abdel-nasser, M., Rashwan, H., Puig, D., 2021a. Grouped Pointwise Convolutions Significantly Reduces Parameters in EfficientNet. IOS Press. pp. 383–391. doi:10.3233/FAIA210158.

Schuler, J., Romaní, S., Abdel-nasser, M., Rashwan, H., Puig, D., 2021b. Reliable Deep Learning Plant Leaf Disease Classification Based on Light-Chroma Separated Branches. IOS Press. pp. 375–381. doi:10.3233/FAIA210157.

Schuler, J.P.S., 2021. K-cai neural api. URL: https://doi.org/10.5281/zenodo.5810092, doi:10.5281/zenodo.5810092.

Schwarz Schuler, J.P., Also, S.R., Puig, D., Rashwan, H., Abdel-Nasser, M., 2022a. An enhanced scheme for reducing the complexity of pointwise convolutions in cnns for image classification based on interleaved grouped filters without divisibility constraints. Entropy 24. URL: https://www.mdpi.com/1099-4300/24/9/1264, doi:10.3390/e24091264.

Schwarz Schuler, J.P., Romani, S., Abdel-Nasser, M., Rashwan, H., Puig, D., 2022b. Color-aware two-branch dcnn for efficient plant disease classification. MENDEL 28, 55–62. URL: https://mendel-journal.org/index.php/mendel/article/view/176, doi:10.13164/mendel.2022.1.055.

Schwarz Schuler, J.P., Romani, S., Abdel-Nasser, M., Rashwan, H., Puig, D., 2022c. Grouped pointwise convolutions reduce parameters in convolutional neural networks. MENDEL 28, 23–31. URL: https://mendel-journal.org/index.php/mendel/article/view/169.

Shafer, S.A., 1992. Using Color to Separate Reflection Components. Jones and Bartlett Publishers, Inc., USA. p. 43–51. URL: https://dl.acm.org/citation.cfm?id=136817#.XEXwBt7cY_w.mendeley.

Shorten, C., Khoshgoftaar, T., 2019. A survey on image data augmentation for deep learning. Journal of Big Data 6. doi:10.1186/s40537-019-0197-0.

Simonyan, K., Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition, in: Bengio, Y., LeCun, Y. (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. URL: http://arxiv.org/abs/1409.1556.

Singh, D., Jain, N., Jain, P., Kayal, P., Kumawat, S., Batra, N., 2020. Plantdoc: A dataset for visual plant disease detection, in: Proceedings of the 7th ACM IKDD CoDS and 25th COMAD, Association for Computing Machinery, New York, NY, USA. p. 249–253. URL: https://doi.org/10.1145/3371158.3371196, doi:10.1145/3371158.3371196.

Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., Stefanovic, D., 2016. Deep neural networks based recognition of plant diseases by leaf image classification. Computational Intelligence and Neuroscience 2016, 3289801. URL: https://doi.org/10.1155/2016/3289801, doi:10.1155/2016/3289801.

Smith, A.R., 1978. Color gamut transform pairs, in: Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques, Association for

Computing Machinery, New York, NY, USA. p. 12–19. URL: https://doi.org/10.1145/800248.807361, doi:10.1145/800248.807361.

Smith, L.N., 2017. Cyclical learning rates for training neural networks, in: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 464–472. doi:10.1109/WACV.2017.58.

Sutherland, A., 1982. On the inherent noise of an ideal two-port isolator. IEEE Transactions on Microwave Theory and Techniques 30, 831–832. doi:10.1109/TMTT.1982.1131149.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Los Alamitos, CA, USA. pp. 1–9. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR.2015.7298594, doi:10.1109/CVPR.2015.7298594.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2016. Rethinking the inception architecture for computer vision, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE. pp. 2818–2826. doi:10.1109/CVPR.2016.308.

Tan, M., Le, Q., 2019. Efficientnet: Rethinking model scaling for convolutional neural networks 97, 6105–6114. URL: https://proceedings.mlr.press/v97/tan19a.html.

Toda, Y., Okura, F., 2019. How convolutional neural networks diagnose plant disease. Plant Phenomics 2019. doi:10.1155/2019/9237136.

Wang, G., Sun, Y., Wang, J., 2017. Automatic image-based plant disease severity estimation using deep learning. Computational Intelligence and Neuroscience 2017, 2917536. URL: https://doi.org/10.1155/2017/2917536, doi:10.1155/2017/2917536.

Wang, M., 2015. Multi-path convolutional neural networks for complex image classification. CoRR abs/1506.04701. URL: http://arxiv.org/abs/1506.04701, arXiv:1506.04701.

Wang, X., Kan, M., Shan, S., Chen, X., 2019. Fully learnable group convolution for acceleration of deep neural networks, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Los Alamitos, CA, USA. pp. 9041–9050. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR.2019.00926, doi:10.1109/CVPR.2019.00926.

Weiss, K., Khoshgoftaar, T.M., Wang, D., 2016. A survey of transfer learning. Journal of Big Data 3, 9. URL: https://doi.org/10.1186/s40537-016-0043-6, doi:10.1186/s40537-016-0043-6.

Wyszecki, G., Wyszecki, G., Stiles, W., Sons, J.W.., 1982. Color Science: Concepts and Methods, Quantitative Data and Formulae. A Wiley-Interscience publication, Wiley. URL: https://books.google.es/books?id=HkjM5HNB6jIC.

Xie, S., Girshick, R.B., Dollár, P., Tu, Z., He, K., 2017. Aggregated residual transformations for deep neural networks, pp. 5987–5995.

Yang, W., Jin, L., Sile, W., Cui, Z., Chen, X., Chen, L., 2019. Thinning of convolutional neural network with mixed pruning. IET Image Processing 13. doi:10.1049/iet-ipr.2018.6191.

Zambon, I., Cecchini, M., Egidi, G., Saporito, M.G., Colantoni, A., 2019. Revolution 4.0: Industry vs. agriculture in a future development for smes. Processes 7, 36.

Zeiler, Matthew D.and Fergus, R., 2014. Visualizing and understanding convolutional networks, in: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (Eds.), Computer Vision – ECCV 2014, Springer International Publishing, Cham. pp. 818–833.

Zhang, T., Qi, G., Xiao, B., Wang, J., 2017. Interleaved group convolutions for deep neural networks. CoRR abs/1707.02725. URL: http://arxiv.org/abs/1707.02725, arXiv:1707.02725.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A., 2016. Learning deep features for discriminative localization, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2921–2929. doi:10.1109/CVPR.2016.319.

Zhuang, Z., Tan, M., Zhuang, B., Liu, J., Guo, Y., Wu, Q., Huang, J., Zhu, J., 2018. Discrimination-aware channel pruning for deep neural networks, in: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 31. Curran Associates, Inc., pp. 881–892. URL: http://papers.nips.cc/paper/7367-discrimination-aware-channel-pruning-for-deep-neural-networks.pdf.

UNIVERSITAT ROVIRA I VIRGILI
EFFECTIVE APPROACHES FOR IMPROVING THE EFFICIENCY OF DEEP CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE CLASSIFICATION
Joao Paulo Schwarz Schuler

UNIVERSITAT
ROVIRA i VIRGILI