Heaven's Light Is Our Guide

AN EFFICIENT IMAGE CLASSIFICATION METHOD USING DEPTHWISE SEPARABLE CONVOLUTION BY A SUPERVISED LEARNING APPROACH

A Thesis Submitted in Partial
fulfillment for the requirement of the
degree of

Bachelor of Science
in
Electrical & Computer Engineering

by

MD Safayet Islam
Roll No. 1610039

to the

Department of Electrical & Computer Engineering
Rajshahi University of Engineering & Technology

October, 2022

Acknowledgement

This thesis has been submitted to the Department of Electrical and Computer Engineering of Rajshahi University of Engineering & Technology (RUET), Rajshahi-6204, Bangladesh, for the partial fulfillment of the requirements for the degree of B.Sc. in Electrical & Computer Engineering. Thesis title regards to "**AN EFFICIENT IMAGE CLASSIFICATION METHOD USING DEPTHWISE SEPARABLE CONVOLUTION BY A SUPERVISED LEARNING APPROACH**".

First and foremost, I offer my sincere gratitude and indebtness to my thesis supervisor, **Rakibul Hassan**, Assistant Professor, Department of Electrical & Computer Engineering who has supported me throughout my thesis with his patience and knowledge. I shall ever remain grateful to him for his valuable guidance, advice, encouragement, cordial and amiable contribution to my thesis.

I wish to thank once again **Prof. Dr. Md. Shahidul Islam** as the head of the Department of Electrical & Computer Engineering for his support and encouragement and also for providing all kind of laboratory facilities.

Finally, I want to thank the most important and the closest persons of my life and my parents for giving a big support to me.

MD Safayet Islam
Roll No. 1610039

October, 2022
RUET, Rajshahi

CERTIFICATE

This is to certify that the thesis entitled " ***AN EFFICIENT IMAGE CLASSIFICATION METHOD USING DEPTHWISE SEPARABLE CONVOLUTION BY A SUPERVISED LEARNING APPROACH*** " by MD Safayet Islam (Roll No. 16610039), has been carried out under my direct supervision. To the best of my knowledge, this thesis is an original one and has not been submitted anywhere for any degree or diploma.

Thesis Supervisor:

.....
Rakibul Hassan
Assistant Professor
Department of Electrical & Computer Engineering
Rajshahi University of Engineering & Technology

CERTIFICATE

This is to certify that the thesis entitled "***AN EFFICIENT IMAGE CLASSIFICATION METHOD USING DEPTHWISE SEPARABLE CONVOLUTION BY A SUPERVISED LEARNING APPROACH*** " has been corrected according to my suggestion and guidance as an external. The quality of the thesis is satisfactory.

External Member:

.....
[**External Member (Bold)**]
[Designation]
Department of Electrical & Computer Engineering
Rajshahi University of Engineering & Technology

Abstract

One of the most well-known areas of machine learning is deep learning. Image classification, which is widely used in computer vision, is one of the essential applications of deep learning. Image classification models have been widely used in a variety of fields, from medicine to autonomous driving. Therefore, a computational and data-efficient model is definitely needed for the real-time classification of objects. In this thesis, we suggest a new CNN architecture for image classification using supervised learning. The utilization of the depthwise and pointwise convolution layers of convolution has entailed some changes. To improve the performance of the depthwise convolution layer, we used two additional 1*1 convolution layers for the reduction of an image's input channels. Additionally, we mixed the spatial dimensions and the reduced channel dimensions using a patch-based representation of the input image with convolution. The residual connections help us to build a deep architecture. We have proposed three different models that are distinguished based on the trainable parameters and training time of the model, keeping in mind the tradeoff between accuracy and speed of training a neural network. To assess our model's ability to classify images, we used the CIFAR10 dataset. We have achieved some competitive results after experimenting with our models. On the CIFAR10 dataset, our smallest model, with only 130,442 trainable parameters and 0.0325 GFLOPS, have a top-1 accuracy of 93%, while our largest model, with 998,666 parameters and 1.0092 GFLOPS, had an accuracy of 97%.

Contents

List of Figures

List of Tables

List of Abbreviations

Short Form	Abbreviations
CNN	Convolutional Neural Network
MLP	Multi-Layer Perceptron
FCN	Fully Connected Network
ANN	Artificial Neural Network
MSE	Mean Squared Error
MAE	Mean Absolute Error
BCE	Binary Cross-Entropy
CCE	Categorical Cross-Entropy
CE	Categorical Cross-Entropy
AI	Artificial Intelligence
ResNet	Residual Neural Network
VGG	Visual Geometry Group
ViT	Vision Transformer
ConvNet	Convolutional Neural Network
CIFAR	Canadian Institute for Advanced Research
KNN	K-Nearest Neighbor
SVM	Support Vector Machine
PIL	Python Image Library
ReLU	Rectified Linear Units
GELU	Gaussian Error Linear Units
SGD	Stochastic gradient descent
FLOPS	Floating Point Operations per Second
TPU	Tensor Processing Unit

Chapter 1

Introduction

Image classification, object identification, image segmentation, and localization are examples of major applications in computer vision. Among these, image classification is the most fundamental challenge. It serves as the foundation for various computer vision challenges. Medical imaging, object recognition in satellite images, traffic control systems, brake light detection, machine vision, and other applications use image classification applications. The procedure of categorizing and identifying sets of pixels or vectors inside an image in accordance with predetermined rules is known as image classification. It is possible to develop the classification legislation using one or more spectral or morphological properties. "Supervised" and "unsupervised" learning techniques are two common types. There is a need for an image classifier that is not only accurate at identifying objects but also light-weight, computationally not heavy, due to the sizeable application space of an image classifier.

1.1 Motivation

Convolutional neural networks have become the most used machine learning techniques for the recognition of visual objects. Though they were first introduced more than 20 years ago, it wasn't until recently that advancements in computer hardware and network architecture made it possible to train truly deep CNNs. VGG[1] had 19 layers, the original LeNet[2] had 5, and recently Residual Networks (ResNets)[3] broke the 100-layer barrier. The size of the parameters and the volume of training data has increased in tandem with the adoption of Transformers[7] as the benchmark for language processing and their improvements in computer vision. Recent research has demonstrated that models based on transformers, most prominently the Vision Transformer (ViT)[4], would perform better in particular circumstances. More crucially, pre-trained on the huge JFT-300M dataset with weak labels,

ViT produces results in comparison to those of state-of-the-art (SOTA) ConvNets, suggesting that Transformer models may be more scale-efficient than ConvNets.

Nevertheless, ViTs cannot be extended to bigger dimensions through the use of patch embeddings, which combine small portions of the picture into a single input element. This is because the self-attention stages in Transformers have quadratic complexity. Additionally, convolutional networks (ConvNets) may exhibit some beneficial inductive biases that plain Transformer layers may not, necessitating the use of a large quantity of data and computer power to make up for this. It should come as no surprise that a lot of recent research has been attempting to integrate the inductive biases of Convolutional networks into Transformer based architecture by attempting to impose local receptive fields for attention layers or enhancing the attention and Feed Forward Network layers with definitive or implicit convolutional operational processes.

These approaches, however, are either ad hoc or concentrated on infusing a specific characteristic, and therefore lack a comprehensive knowledge of the relative functions of convolution and attention while coupled. This trend raises serious issues, notably, but not limited to, the data shortage in some fields of science and the isolation of individuals with limited financial resources from research work. It is difficult for everyone to possess a large dataset or have large computing resources due to the fact that computer vision is being employed in many different sectors and on many different platforms. Therefore, the requirement for an image classifier with lower complexity and acceptable accuracy exists. In this paper, we looked into patch-based image representation using depth-separable convolution for image classification.

To create a novel image classification architecture with fewer parameters and better accuracy, we experimented with the usage of Depthwise Separable Convolution[5] in this study.

1.2 Literature Review

1.2.1 Case 1

Krizhevsky et al. [6] have suggested using five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers as ConvNet architecture. To incorporate non-linearity, the suggested model employed Rectified Linear Activation (ReLU) Activation Function rather than Tanh activation. To deal

with overfitting training data, the dropout regularization technique was also applied. The network size was decreased by using the Max Pooling Layer. This model has relatively little depth, which makes it difficult for it to learn features from image sets. There are just eight learnable layers in it. Compared to contemporary ConvNet models, it takes longer to obtain results with higher accuracy. The network design has 62.3 million learnable parameters, which is a very high quantity. This model's accuracy on the CIFAR10 dataset is about 80.80%.

1.2.2 Case 2

Simonyan et al. [1] suggested using a small (3*3) filter size for convolution layers to increase the depth of the CNN. They suggested other CNN models of various sizes built on the same architecture. VGG-16 has 16 layers, and VGG-19 has a depth of 19 layers. In order to improve the performance of the model, VGG was built.

Non-linearity increased as the number of layers with smaller kernels increased. The Vanishing Gradient issue exists in that proposed CNN paradigm. Compared to the more recent ResNet Architecture, VGG-19 takes a very long time to train. The network design weighs in at 549 MB, which is quite a bit. The network design features 143.7 Million Learnable Parameters, which is a very high quantity. The model's accuracy on the CIFAR10 dataset is 91.2%.

1.2.3 Case 3

He et al. [3] have recommended employing skip connections to deepen a network while avoiding vanishing gradient issues. By including residual Block, they were able to address the Deep Neural Network degradation issue. By doing so, they were able to create a model that was considerably deeper and didn't have the exploding/vanishing gradient problem. To reduce overfitting, they utilized batch normalization. The complexity of the architecture was boosted by the deep model. Later, it was discovered that accuracy reached saturation with an increase in data. The ResNet-50 model on an NVIDIA M40 GPU requires 14 days to train to complete 90 epochs on the ImageNet dataset. There are 25.6 million learnable parameters in the network design. On the CIFAR10 dataset, the model's accuracy is 95.84%.

1.2.4 Case 4

Dosovitskiy et al. [4] proposed the use of Self-Attention instead of Convolution layers for image classification. The various components of an image are represented via patch embedding. No convolution was utilized, hence there was no bias in the architecture that was image-specific. With more data available, they observed an improvement in the model's accuracy. However, when the model was trained on insufficient amounts of data, it did not generalize effectively. For the model to achieve state-of-the-art accuracy, 300M pictures were used in training. Large Model Size (relative to ResNet, 307M parameters) (0.85M). Large CO2 footprint because it requires about 30 days to train a ViT on an 8-core, typical cloud TPUv3. On the CIFAR10 dataset, the model's accuracy is around 90%.

1.3 Thesis Objective

- Select a suitable dataset to evaluate the effectiveness and accuracy of the suggested model.
- Create a new architecture for deep learning-based image classification.
- Reduce the model's parameters while maintaining accuracy to improve it.
- Reduce the computational complexity of the model.
- To analyze the impact of different hyperparameters on the output of the model.
- To analyze the impact of different optimizers, loss functions, and activation functions.
- Use various state-of-the-art training paradigms to get the highest possible accuracy.
- Compare the proposed model with other deep learning image classification models.
- Obtain classification reports for diverse datasets of images from different domains.
- Reduce the training time of the model using various techniques.

1.4 Thesis Outline

- The research work is introduced in Chapter 1 along with the rationale behind it. Current research in this topic is also covered, along with its limitations. Here, the research's goals have also been listed.
- Chapter 2 provides information on the dataset that was used to evaluate the performance of the model, various data preprocessing and feature extraction approaches, deep learning layers, and model-building functions.
- The details of our experimental findings, the impact of various hyperparameters on the model's overall performance, and comparisons with other models are covered in Chapter 3.
- The summary of our research, as well as the limitations and future directions of our work, are concluded in Chapter 4.

Chapter 2

Methodology

To improve the accuracy of our model, we adhered to a few well-suited methodologies that were most appropriate for our research. This includes picking the right dataset, applying the right preprocessing and augmentation methods, and using the right layers, kernels, and convolution layer filter sizes. We also adhered to standard practices when training a deep learning model, such as comparing various learning rate schedulers and selecting an appropriate optimizer after careful consideration. Due to all of these choices, we can obtain a competitive accuracy on the specified dataset with fewer training parameters and a simpler computational requirement.

2.1 Image Classification Techniques

The field of image classification was founded in the 1960s with the audacious objective of emulating human vision systems. The aim was overly wide, and the level of computing at the time was fairly low. After that, a long time had gone. The decade of the 2010s was the apex of image classification and all of the artificial intelligence. The Imagenet [8] Challenge, a massive collection of annotated images with over a million images and thousands of labels, helped this topic gain new popularity. This was the first time deep neural networks were trained on such a large dataset using GPUs to identify images; as a result, AlexNet [6] won the first Imagenet competition with a top-5 error rate of 15.3%, 10.8% lower than the runner-up. The foundation of algorithms used in computer vision is still Yann Le Cunn's convolutional neural network [2]. Inception [9] networks developed by Google introduced a novel idea of stacking convolutions with various filter sizes that processed the same input and outperformed Alexnet on the ImageNet task. Then came algorithms like Resnet [3], VGG-16 [1], and many others that outperformed Alexnet [6]. We have advanced to the point where, on numerous datasets, image classification models outperform human baseline performance. Therefore, understanding the contextual

information in images in order to categorize them into a set of predefined labels is likely one of the most fundamental and essential tasks in computer vision.

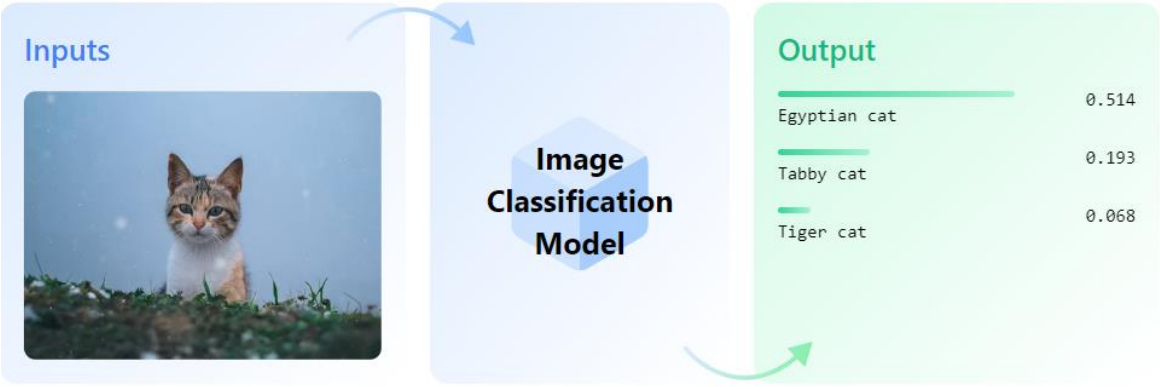

Figure 2.1: Image Classification Basics [25].

2.2 Building Blocks for Image Classification

- **Image Pre-processing**: By reducing undesirable distortions and enhancing a few key image qualities, this approach tries to enhance the visual information (features) so that machine learning models can use it to their advantage. Pre-processing for images includes reading the original image, scaling the picture, and data augmentation. There are many strategies for augmentation, including grayscale image resizing, histogram equalization, contrast stretching, and changing the saturation, hue, and brightness, among others.
- **Object Detection**: The term "detection" describes the positioning of an object, which entails segmenting the image and locating the target object of particular interest.
- **Extracting Features and Training Procedure:** This is indeed a vital step when the machine or deep learning algorithms are used to find the most intriguing patterns in the picture, characteristics that may be specific to a certain class and that will subsequently aid the model in differentiating between classes. Model training refers to the procedure wherein the model picks up a set of features from the feature set.
- **Object Classification:** In this step, identified items are categorized into specified classes using an appropriate classification algorithm that contrasts the desired characteristics with the visual features.

2.3 Classification Method

2.3.1 Supervised Classification

The principle behind supervised classification is that a user can choose a sample of pixels from an image that best represents different classes, and can then instruct the image processing system to utilize these training sites as a guide for classifying all those other pixels in the image. The user's knowledge is taken into consideration while choosing training sites, sometimes referred to as training images or input classes. The threshold for how comparable adjacent pixels must be grouped is likewise chosen by the user. These limits are frequently established using the training area's spectral properties. The user can additionally choose how many classes an image will be divided into. After each information class has been statistically characterized, the image is subsequently categorized by determining which of the characteristics each pixel's reflectance most closely resembles. To create predictive models, supervised classification employs regression techniques and classification algorithms. The methods are K-Nearest Neighbor(KNN), Naive Bayes classifiers, Decision Trees, Support Vector Machines(SVM), and Artificial Neural Networks(ANN), Convolutional Neural Networks(CNN).

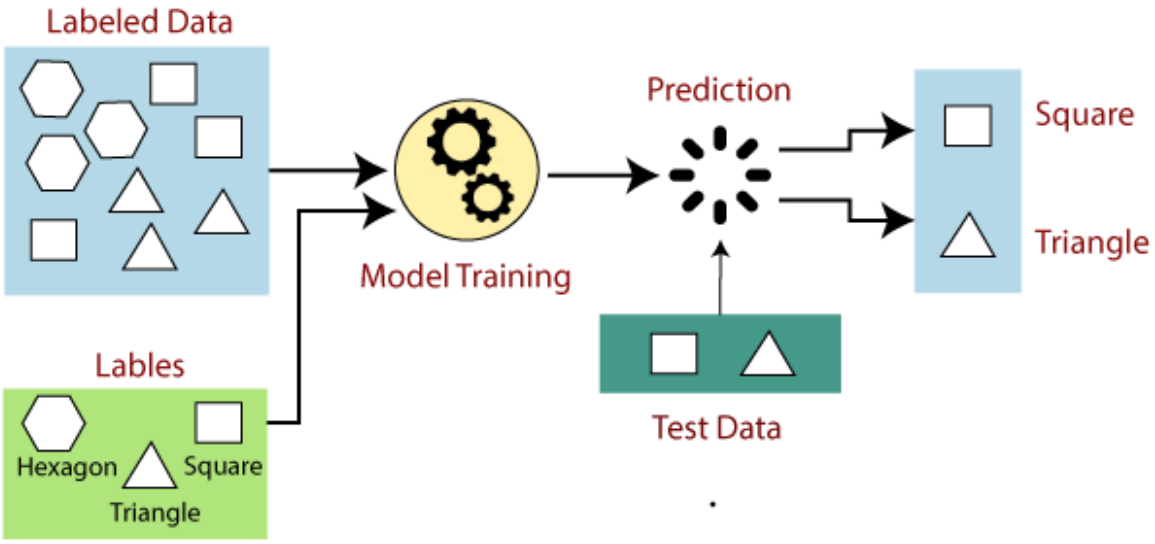

Figure 2.2: Supervised Classification [26].

2.4 Dataset

The performance of the proposed model is assessed using the CIFAR10 [10] benchmark dataset, which was created by Krizhevsky et al. to test the performance of an image classifier. He et al. [3] and Krizhevsky et al. [1] have both used this dataset in their studies. It has 6000 images in each class and a total of 60000 32x32 RGB color images organized into 10 classes. 10000 test images and 50,000 training images are available in the dataset.

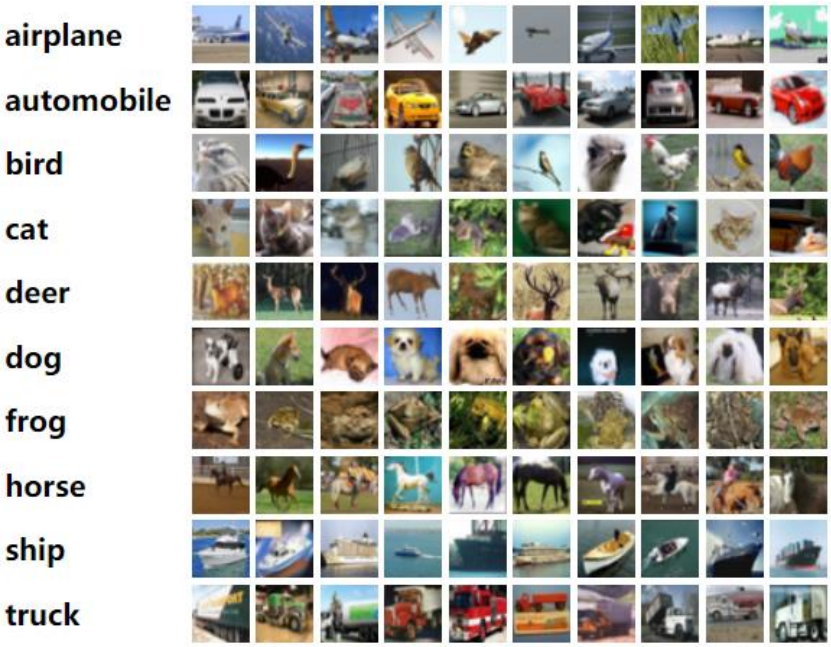

Figure 2.3: Overview of CIFAR10 Dataset [10].

2.4.1 Dataset Information

A subset of the Tiny Images collection, the CIFAR-10 [10] dataset (Canadian Institute for Advanced Research, 10 classes) contains 60000 32x32 color images. The images are assigned to one of ten mutually exclusive classes, including airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. 5000 training photos and 1000 testing images make up each of the 6000 images in a class.

The following criteria were used to determine whether an image belonged in a class:

- When asked, "What is in this picture?," the class name ought to be near the top of the list of likely responses.

- The picture must be realistic in every way. Line drawings should not be accepted, according to labelers.
- Only one visible version of the object that the class refers to should be present in the image. If the labeler can still tell what the object is, it can be partially obscured or seen from an unexpected angle.

Table 2.1: Class Level and Description of CIFAR10 Dataset.

Label	Description
0	airplane
1	automobile
2	bird
3	cat
4	deer
5	dog
6	frog
7	horse
8	ship
9	truck

2.4.2 Dataset Preprocessing

Grayscale conversion: Grayscale is just the process of turning colored images into black and white. It is typically utilized to make machine learning methods less computationally complex. Grayscale is a smart choice because it minimizes the number of pixels found in an image, which lowers the number of computations needed, and most photographs don't need a color scheme to be recognized.

Normalization: The technique of projecting picture information pixels (intensity) to a preset range, typically (0,1) or (-1,1), is known as normalization or data re-scaling. This is frequently applied to various data formats, thus we would like to standardize them all so that we can use the same algorithms on them all. Normalization is typically used to translate the pixel values of a picture into a more common or comfortable perception.

Benefits of it include:

1. Fairness across all photos - For instance, scaling every picture to an equal [0, 1] or [-1, 1] range enables all pictures to make contributions to the overall loss as opposed to that when some photos have both high and low pixel value ranging to give strong and inadequate loss, respectively.
2. Re-scaling helps to offer a standard learning proportion across all pictures because high-pixel images need a small learning rate and low-pixel images need a high learning rate.

Data Augmentation: Making small changes to already-existing data in order to broaden its diversity without gathering new data is known as data augmentation.
It is a method for increasing a dataset. Standard data augmentation methods include flipping data horizontally and vertically, rotating data, cropping data, shearing data, etc.
Data augmentation can assist stop a neural net from picking up unrelated features. As a result, the model performs better.
Standard data augmentation methods include flipping data horizontally and vertically, rotating data, cropping data, shearing data, etc.
Two categories of augmentation exist:

1. For tiny datasets, offline augmentation is used. It is used in the preliminary data processing stage.
2. 2. For large datasets, online augmentation is used. Usually, it is used in real-time.

As our dataset consists of 60,000 images, we have used real time data augmentation techniques to prevent overfitting. We have selected the most popular techniques like AutoAugmentation, RandAugment, CutMix, Mixup to make the classifier more robust.

AutoAugmentation: Data augmentation policies can be discovered automatically using AutoAugment [11]. Finding the ideal augmentation policy is formulated as a discrete complex problem. It has two parts: a search space and a search algorithm. To a greater extent, the search algorithm (expressed as a Recurrent Neural Network) selects an image augmentation policy **S** that contains data on which image analysis operation to use, how likely it is to be used in each batch, and how much of the operation to utilize. A neural net with a predefined architecture is trained using the policy **S**, and the verification

correctness **R** is then delivered back to change the controller. **R** cannot be differentiated; hence policy gradient methods will be used to update the controller. All PIL functions that require a picture as an input and produce an image are used in the operations. PIL is a well-known Python image library. Cutout and SamplePairing are two further augmentation methods used. Shear X/Y, Translate X/Y, Rotate, Auto Contrast, Transpose, Normalize, Solarize, Posterize, Contrast, Hue, Luminance, Sharpness, Cutout, and Sample Pairing are the operations that were looked up.

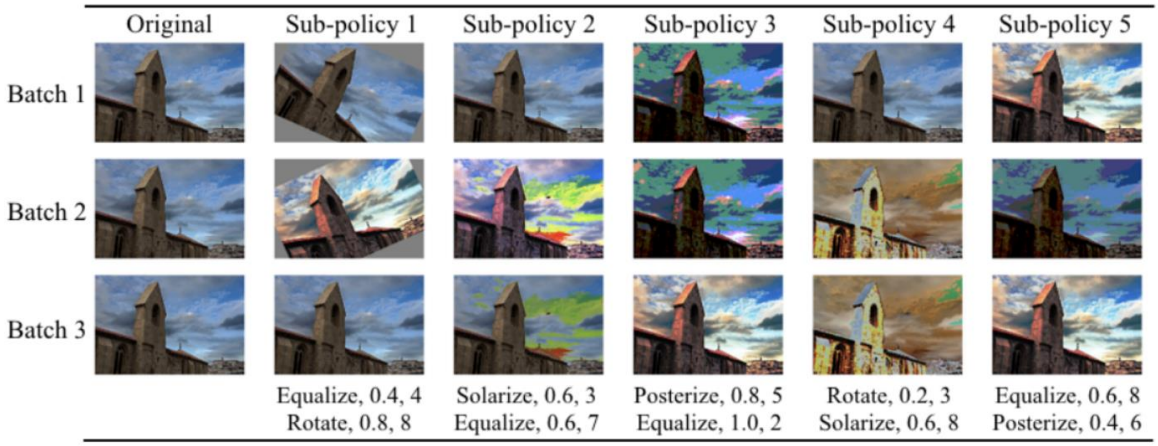

Figure 2.4: Image Augmented by of AutoAugmentation [11].

RandAugment: An automated image augmentation technique is RandAugment [12]. Two interpretable hyperparameters, **N** and **M**, are present in the problem space for data augmentation. The frequency of augmentation adjustments to be applied in succession is N, and their combined magnitude is **M**. The learned rules and probabilities for executing every alteration are substituted with a parameter-free technique that always chooses a conversion with uniform probability **1/K**, which reduces the parameter space while maintaining image diversity. **K** indicates that there are **K** different ways to change. RandAugment can express **KN** possible policies given N modifications for a training image.

Among the transformations utilized are identity transformation, auto contrast, normalization, rotation, solarization, color jittering, posterization, varying brightness, varying clarity, shear-x, shear-y, and translate-x, translate-y.

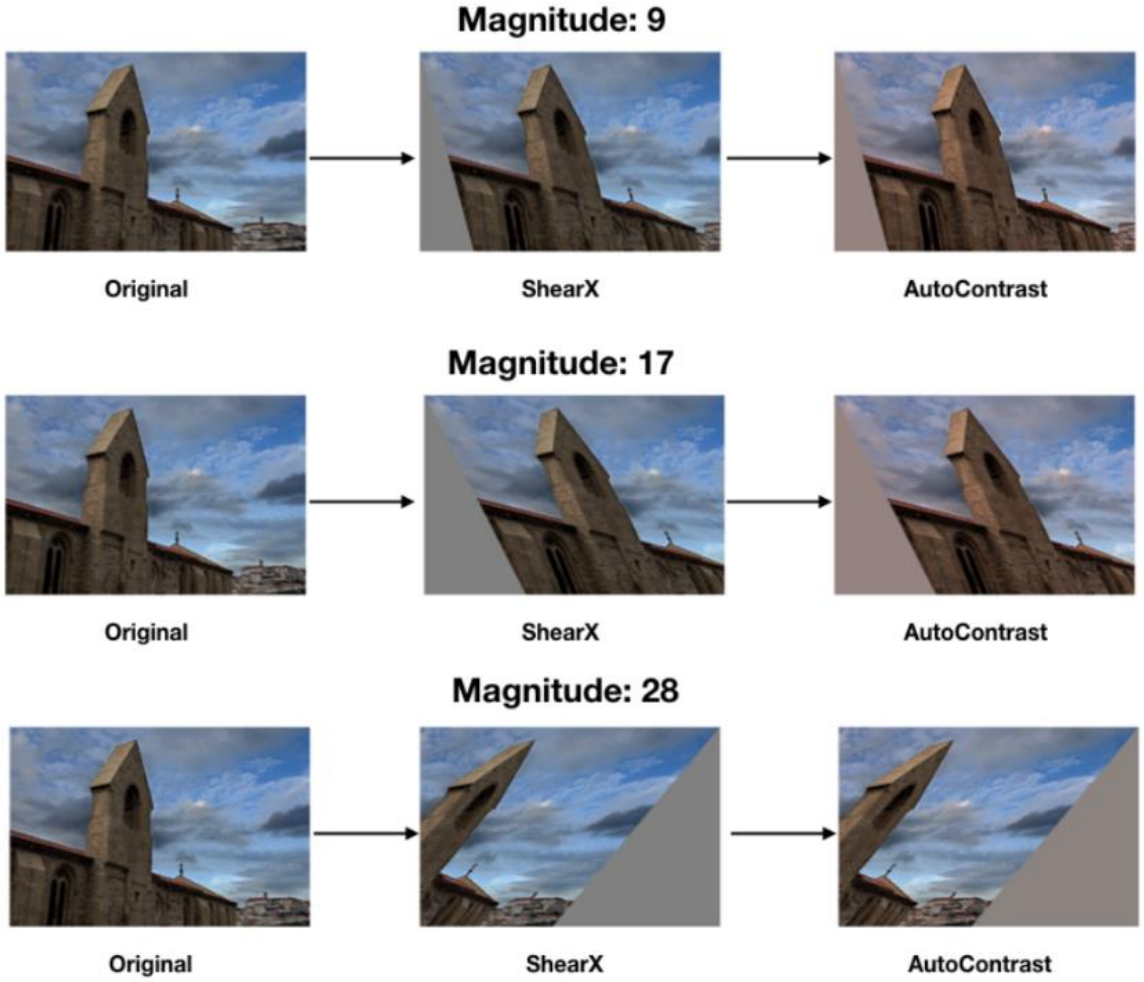

Figure 2.5: Image Augmented by of RandAugment [12].

CutMix: CutMix [13] is a technique for enhancing visual data. Unlike Cutout, we substitute the removed portions with a patch from some other image instead of just eliminating pixels as in Cutout. In accordance with the total number of pixels in the merged images, the ground truth values are also mixed. The additional patches make the model recognize the item from a partial perspective, which improves localization capability even further.

Figure 2.6: Image Augmented by of CutMix [27].

MixUp: Using the training data, the data augmentation method known as MixUp [22] creates a combination of the probability of random image pairs. Given two images and their ground truth labels: (x_i, y_i), (x_j, y_j), a synthetic training example $(\hat{x}, \hat{y})$ is generated as:

$$\hat{x}=\lambda x_i+(1-\lambda)x_j \quad \hat{y}=\lambda y_i+(1-\lambda)y_j$$

where $\lambda \sim \text{Beta}(\alpha=0.2)$ is independently sampled for each augmented example.

2.5 Network Architecture

The building block of our classification model is Convolution layer. We used different Convolution configuration in our model. In addition, we also used Residual Skip Connections to prevent vanishing gradient problem and to help the network converge faster.

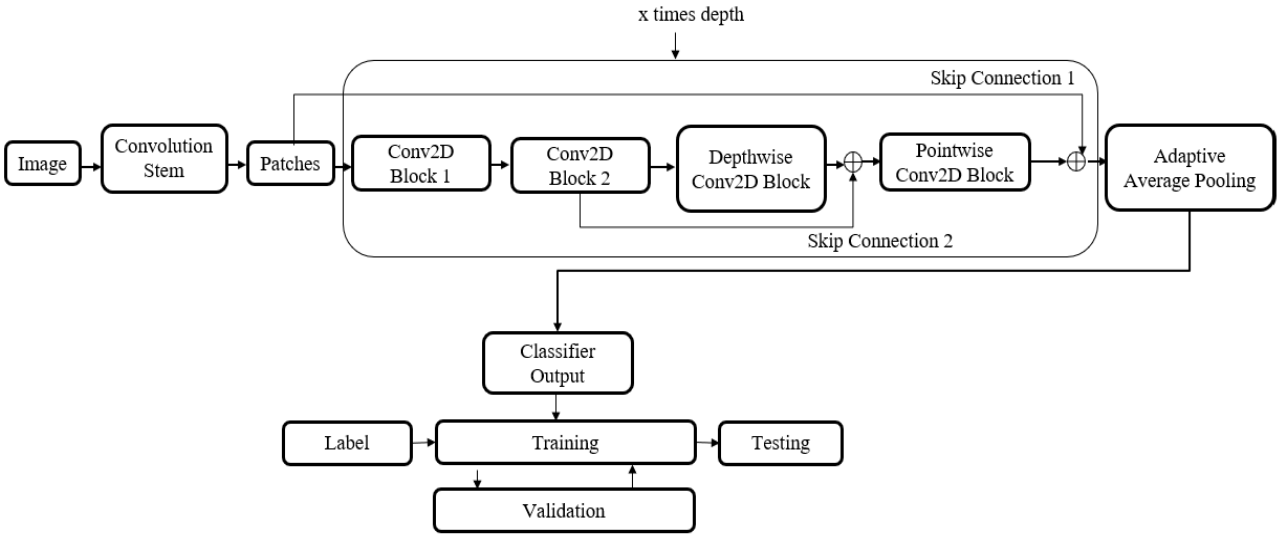

Figure 2.7: Proposed CNN Model Architecture.

2.5.1 Convolution Stem Block: Over the initial image, the stem layer functions as a compression technique. As a result, the spatial extent of the activations is quickly reduced, which lowers memory and computing costs. Patches of the image are then separated. A convolution layer with c input channels and h output channels, kernel size, and stride equal to the patch size make up this layer. The projection dimension of the patches is indicated by the number of filters employed in this layer.

An activation function and a post-activation batch normalization [14] come afterward. The model's initial element is this layer. The output of this layer can be represented as:

$$z0 = BN(\sigma Conv_c \rightarrow_h (X, stride = p, kernelsize = p))$$

2.5.2 Image Patches: The images were divided into several patches that were of the same length and width, **P**.

$$\boldsymbol{x} \in \boldsymbol{R}^{\boldsymbol{H}\times\boldsymbol{W}\times\boldsymbol{C}} \Rightarrow \boldsymbol{x} \in \boldsymbol{R}^{\boldsymbol{N}\times\boldsymbol{P}^{\boldsymbol{2}}\boldsymbol{C}}$$

Where,

- $N = \frac{HW}{P^2}$ is the number of patches
- **P** is the height and width of the patch with a patch dimension (**P, P, C)**
- **C** is the number of channels

The stride of the convolution stem layer works as a hyperparameter as it determines the size of the image patches.

2.5.3 Conv2D Block 1: The stack of convolution layers utilized in each block starts with this layer. This layer serves as a channel downsizing block. The filter size functions as a hyperparameter while the kernel size of the convolution layer is held constant at 1. The multiplication required for the subsequent layers is minimized by narrowing the channel. The kernel size half of the projection dim produces the best results for the CIFAR10 dataset. The convolution layer is followed by the GELU [15] activation layer and then the batch normalization layer.

2.5.4 Conv2D Block 2: The convolution block after that functions as a channel dimensionality reduction method as well. Here, the filter size is set to one-quarter of the projection dim and the kernel size is left at 1. Similarly, batch normalization and GELU activation are applied after this convolution layer.

2.5.5 Depthwise Convolution Block: Here, Depthwise convolution is performed over the reduced channel. The depthwise convolution layer's kernel size is crucial to the model's accuracy. A bigger kernel size promotes information exchange between adjacent pixels but also raises the cost of calculation. In the suggested architecture, it has been discovered that the kernel sizes of 5 and 7 offer a fair balance between precision and complexity. A residual connection is established between the output of Conv2D block 2 and this output for efficient information transfer after the activation function and batch normalization.

2.5.6 Pointwise Convolution Block: A 1x1 kernel is used in the Pointwise Convolution method, which iterates over each and every point. The depth of this kernel is the same as the number of channels in the input image. To construct depthwise-separable convolutions, it is combined with depthwise convolutions. The filter size in this case is maintained at the same level as the patch projection dimension. Another residual link is formed between the input of Conv2D block 1 and this block's output after activation and normalization have been completed.

2.6 Convolutional Neural Network

Convolutional neural networks outperform other neural networks when given inputs such as images, voice, or audio, for example. There are three basic categories of layers in them:

- Convolutional layer
- Pooling layer
- FC (fully-connected) layer

A convolutional network's first layer is the convolutional layer. The fully-connected layer serves as the final layer, even though convolutional layers, further convolutional layers, or pooling layers, can come after it. The CNN becomes more complicated with each layer, detecting larger areas of the image. Early layers emphasize basic elements like colors and borders. The larger features or shapes of the image are first recognized when the visual data moves through the CNN layers, and eventually the intended object is recognized.

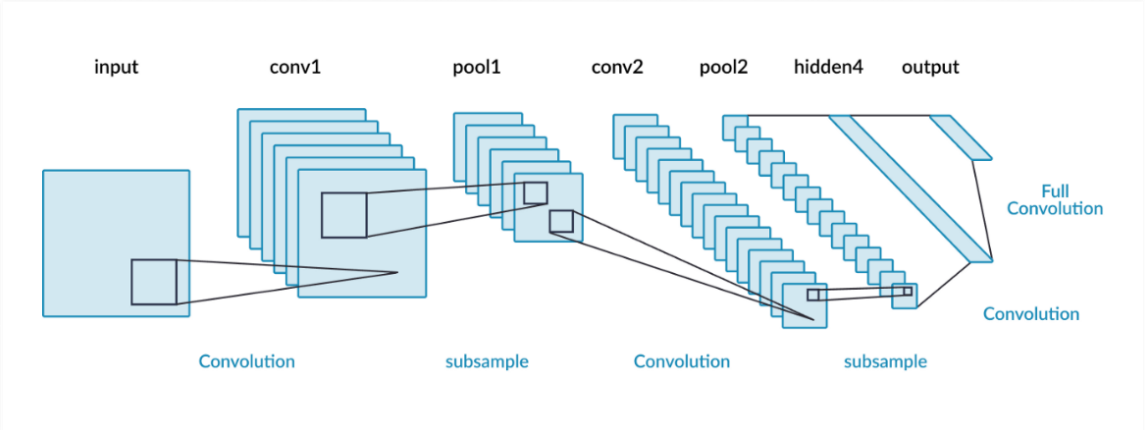

Figure 2.8: Convolutional Neural Network [28].

2.6.1 Convolutional Layer

The central component of a CNN is the convolutional layer, which is also where the majority of the processing takes place. It needs raw data, feature space, and a filter, among other things. Assuming that the input data will consist of a color picture that is composed of a 3D pixel matrix. As a result, the input data will have three dimensions—height, width, and depth—that are analogous to RGB in an image. Additionally, we have a feature extraction technique, also referred to as a kernel or filter, which would move through the image's receptive fields and determine whether the feature is there. Convolution describes this process.

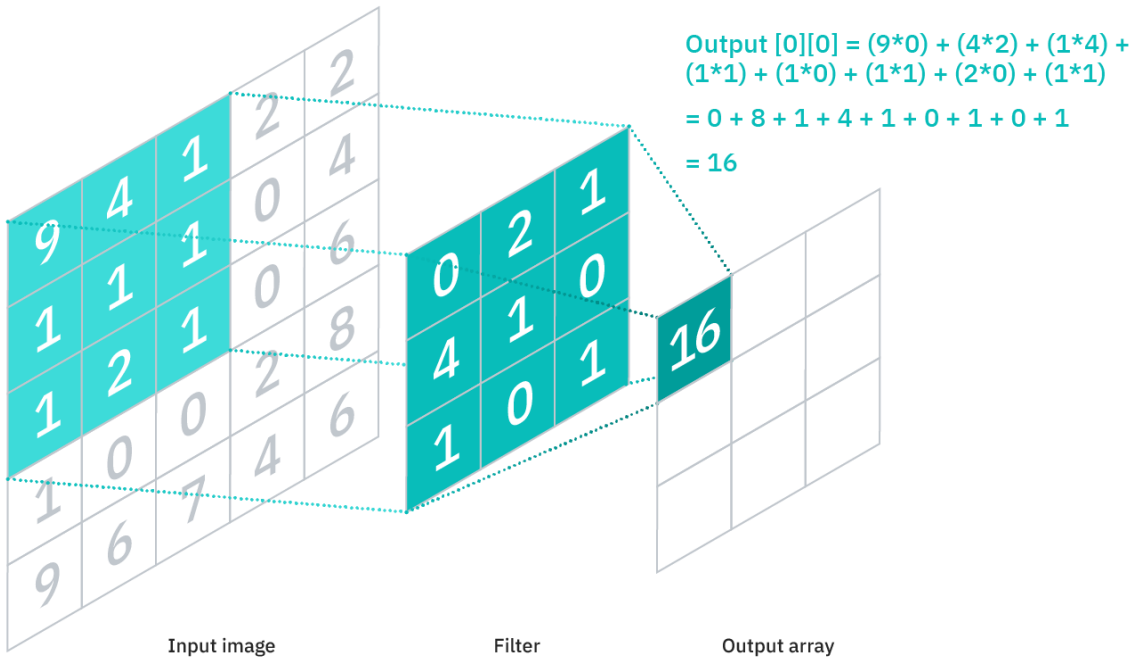

Figure 2.9: Convolution Process [29].

A 2-D weighted array that represents a portion of the image serves as the feature detector. The filter size is normally a 3x3 matrix, however, they can be other sizes; this also influences how big the receptive field will be. Following the application of the filter to a portion of the image, the dot product between the pixel intensities and the filter is determined. The output array is then fed with this dot product. Once the kernel has scanned through the entire image, the filter moves by a stride and repeats the operation. Feature maps, activation maps, and

convolved features are all terms used to describe the final output of the succession of consecutive dot products from the input and filter.

As seen in the aforementioned figure, not every resulting value that lies in the feature space needs to correspond to every intensity value in the input image. It simply needs to be connected to the area where the filter has been applied, or the receptive field. Convolutional layers and also pooling layers are frequently referred to as "partially connected" layers because the output matrix does not have to map exactly to each input value. The local connection is another name for this characteristic. The feature detector, commonly referred to as parameter sharing, traverses over the image while maintaining a fixed set of weights. Through the use of gradient descent and backpropagation, some parameters, such as weight values, adapt during training. Nonetheless, there are three hyperparameters that must be specified before the neural network can start to be trained that have an influence on the output volume size.

These consist of:

1. The output's depth is influenced by the quantity of filters. A depth of three would result from, for instance, the use of three different filters, which would produce three distinct feature sets.

2. The kernel's stride is the amount of pixels it travels over the input vector. Despite the rarity of stride lengths of two or higher, a longer stride results in a lesser output.

3. In cases where the filters would not cover the input image, zero padding is typically utilized. This produces a bigger or identically sized output by setting any elements that are not part of the input vector to zero.

Padding comes in three different forms:

1. Valid padding: Also referred to as zero padding. In this situation, if aspects do not line up, the final convolution is skipped.

2. Same padding: By using this padding, the input layer and output layer will be of equal size.

3. Full padding: By padding the input's border with zeros, this technique expands the output's size.

Rectified Linear Unit (ReLU) [16] activation is applied to the feature map following each convolution operation in a CNN, which adds nonlinearity to the model.

As was previously mentioned, the first convolution layer may be followed by another convolution layer. When this occurs, the CNN's structure may become hierarchical because the later layers will be able to view the pixels in the earlier layers' receptive fields. Let's use the case of trying to determine whether a bicycle is there in an image as an example. The bicycle can be viewed as a collection of components. It is made up of a chassis, handlebar, tires, paddles, and so on. A feature hierarchy is created within the CNN by the bicycle's component pieces, each of which represents a lower-level structure in the neural network and the bicycle as a whole a higher-level structure.

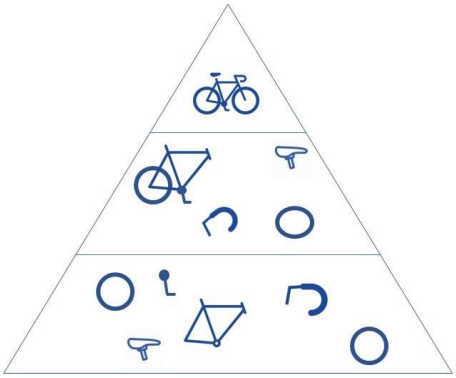

Figure 2.10: Feature Hierarchy in Convolution Process [29].

2.6.2 Pooling Layer

Down sampling, also referred to as pooling layers, does dimension reduction by minimizing the number of components in the data. The pooling process spreads a kernel across the whole input, much like the convolution operation does, however this filter doesn't contain any weights. In contrast, the output array is filled by the kernel using an aggregation function applied to the values in the receptive field.

There are principally two forms of pooling:

1. **Max pooling:** The filter chooses the input pixel with the highest value to transfer to the output vector as it advances across the input. As a side note, this method is applied more frequently than average pooling.

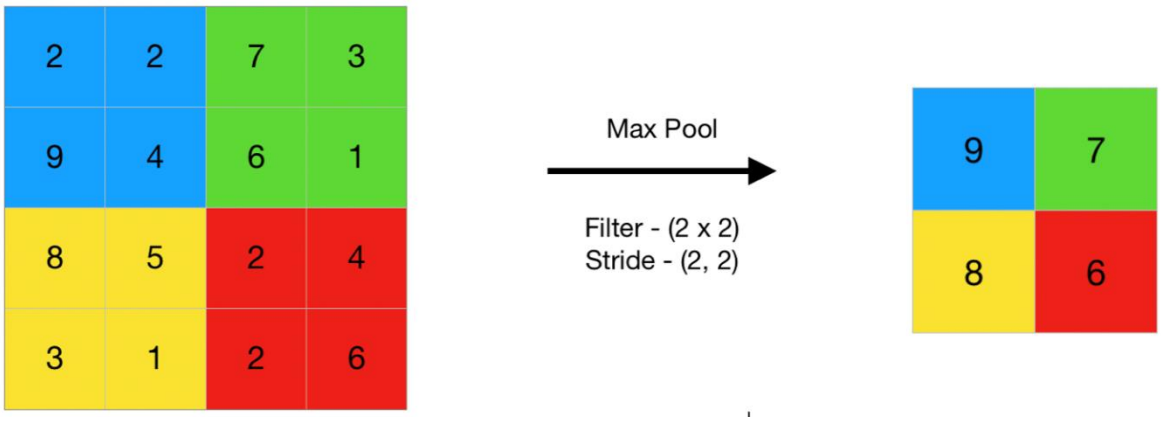

Figure 2.11: Max pooling Operation [30].

2. **Average pooling:** The filter computes the average value inside the receptive field as it advances across the input and sends that value to the output vector.

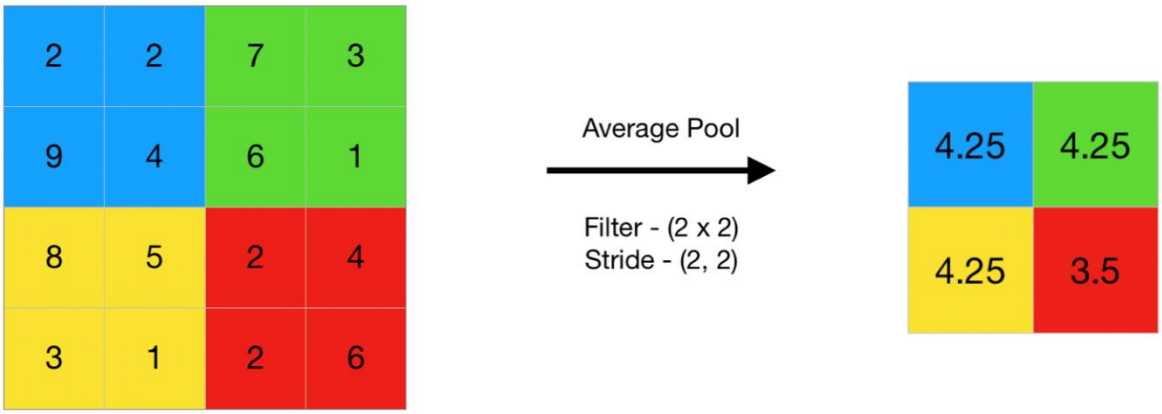

Figure 2.12: Average pooling Operation [30].

Although the pooling layer results in a significant amount of information loss, it also offers CNN a number of advantages. They lessen complexity, increase effectiveness, and lower the risk of overfitting.

2.6.3 Fully-Connected Network

The full-connected layer is exactly what its name implies. As was already noted, partially interconnected layers do not have a direct connection between the input image's pixel values and the output layer. In contrast, every node in the output layer of the fully-connected layer is intrinsically linked to a node in the layer above it. Based on the characteristics that were retrieved from the preceding layers and their various filters, this layer conducts the classification operation. FC layers often utilize a Softmax activation function to categorize inputs adequately, yielding a probability ranging from 0 to 1. Convolutional and pooling layers typically use ReLU activation functions.

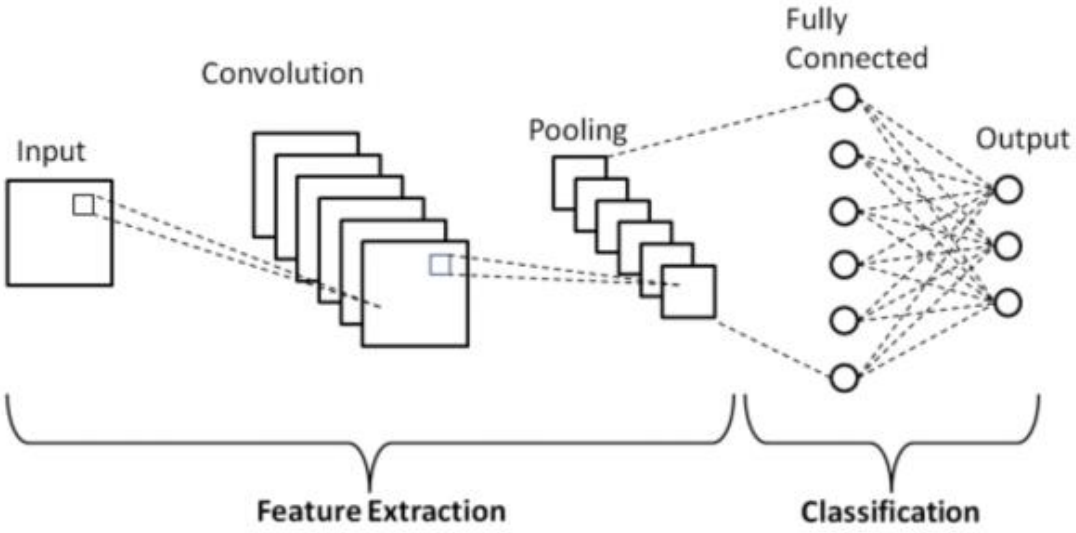

Figure 2.13: Fully Connected Layers in CNN [31].

2.7 Depth-wise Convolution and Depth-wise Separable Convolution

Depthwise Separable Convolution divides the computation into two stages, as compared to conventional convolution, which does the channel wise and spatially-wise processing in one step. A standard CNN filter is applied for each input channel during depthwise convolution, and the output of depthwise convolution is then combined linearly using pointwise convolution. Input*Output*Width*Height parameters—whose width and height correspond to the filter's width and height—are used in a neural network's typical convolution layer. A 7*7 filter with 2800 parameters will contain 2800 parameters for an input node of 10 and an

output channel of 20. The likelihood of over-fitting rises with the number of parameters. Researchers have frequently searched for various convolutions to prevent such situations. Both depth-wise separable convolution and depth-wise convolution fit into those categories [32].

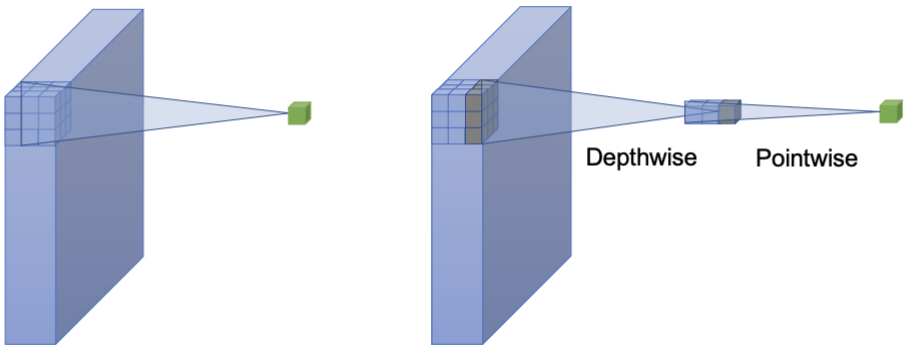

Figure 2.14: Standard Convolution and Depthwise Separable Convolution [32].

2.7.1 Depth-wise convolution

Throughout this convolution, every depth level of a input tensor is subjected to a 2-d depth filter. Let's use an instance to better understand this. Assume that our input tensor has the dimensions 3* 8*8 (input channels*width*height). Its dimensions are 3*3*3. In a typical convolution, the depth dimension would also be directly convolved.

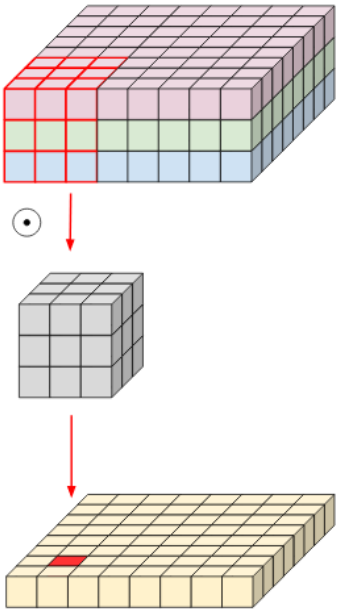

Figure 2.15: Normal Convolution [33].

Each filter channel is only used at one input channel during depth-wise convolution. We have a three-channel filter and three-channel picture in the example. To do this, we separate the filter and picture into three different channels, convolve the associated picture with the affiliated channel, and then stack the results back together.

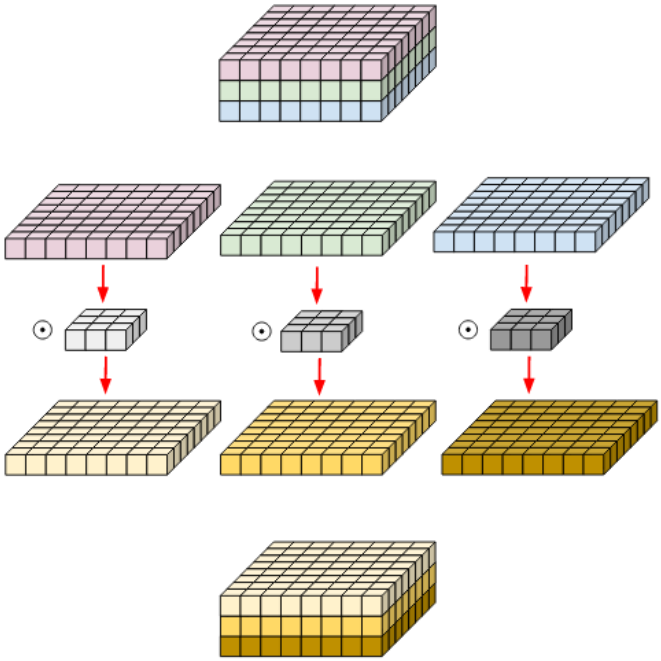

Figure 2.16: Depth-wise convolution with three separate channels [33].

Normal convolution can achieve the same result if we choose a channel, set all of the filter's elements to 0, except for that channel, and afterwards convolve. One filter will be required for each channel, making a total of three. Despite the fact that the parameters are unchanged, this convolution system consists of three output channels using only one 3-channel filter, when standard convolution would necessitate three 3-channel filters.

2.7.2 Depth-wise Separable convolution

The term separable refers to a convolution that was inspired by the notion that a kernel's depth and spatial dimensions may be separated. Let's use the Sobel filter as an example, which is employed in computer vision to identify edges. The measurements of these filters' height and width can be separated. Vector multiplication of [1 2 1] transposed with [-1 0 1] might be thought of as the Gx filter.

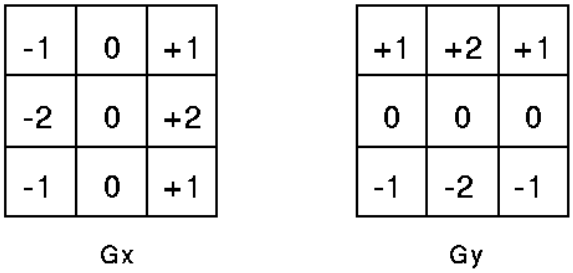

Figure 2.17: Sobel Filter for Vertical & Horizontal Edge Detection [33].

The filter's deception is apparent. It indicates that it had 9 parameters, but only has 6. This has been made possible by the isolation of its own length and breadth dimensions. We may conduct depth-wise convolution and then employ a 1*1 kernel to encompass the depth dimension thanks to the same concept employed to separate the depth dimensions from horizontal (width*height).

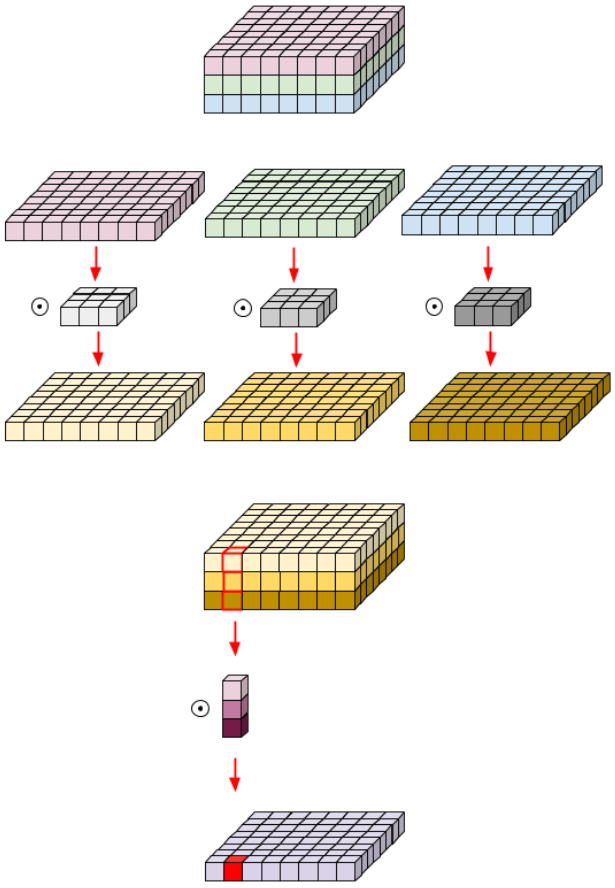

Figure 2.18: Depth-wise Separable Convolution Process [33].

One point to consider is the number of parameters is lowered by such a convolution to produce the same amount of channels. Three three-by-three-by-three parameters are required to execute depth-wise convolution on one channel, and one three-by-three-parameter convolution on the depth dimension. However, if we just want 3 output channels, we simply require 3, 1*3 depth filters, providing us a total of 36 (= 27 + 9) parameters. In contrast, for the identical set of output channels in standard convolution, we only require 3, 3*3*3 filters, providing us with just an aggregate of 81 parameters. When there are excessively many parameters, the function must be memorized rather than learned, which leads to over-fitting. We avoid that with depth-wise and separable convolution.

2.8 Activation Functions

2.8.1 Rectified Linear Units

ReLUs [16], a particular class of activation function, are linear in the positive dimension but zero in the negative. The non-linearity is caused by the function's kink. Contrary to sigmoid activations, linearity in the positive dimension has the desirable virtue of preventing gradient non-saturation, even if its gradient is zero for half of the real line.

$$\boldsymbol{f(x) = \max(0, x)}$$

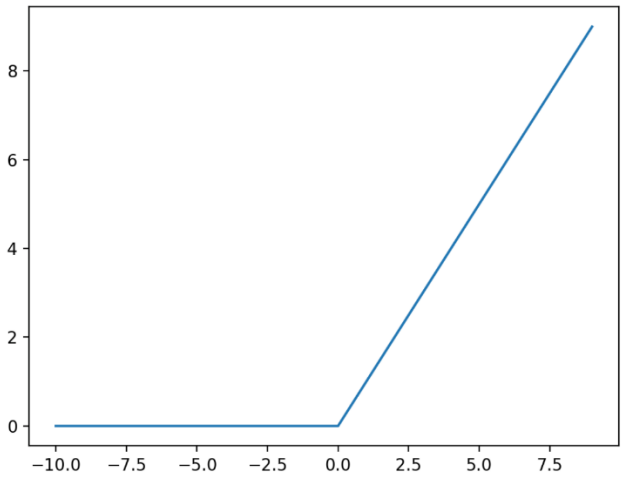

Figure 2.19: Rectified Linear Units [34].

2.8.2 Gaussian Error Linear Units

The GELU [15], also known as the Gaussian Error Linear Unit, is an activation function. The typical Gaussian cumulative distribution function, serves as the basis for the GELU activation function. Instead than weighing inputs by respective signs as in ReLUs. the GELU non - linear weights inputs according to their percentile. As a result, the GELU can always be viewed as a more fluid ReLU.

$$\mathrm{GELU}(x) = xP(X \leq x) = x\Phi(x) = x \cdot \frac{1}{2}\left[1 + \mathrm{erf}\left(x/\sqrt{2}\right)\right],$$

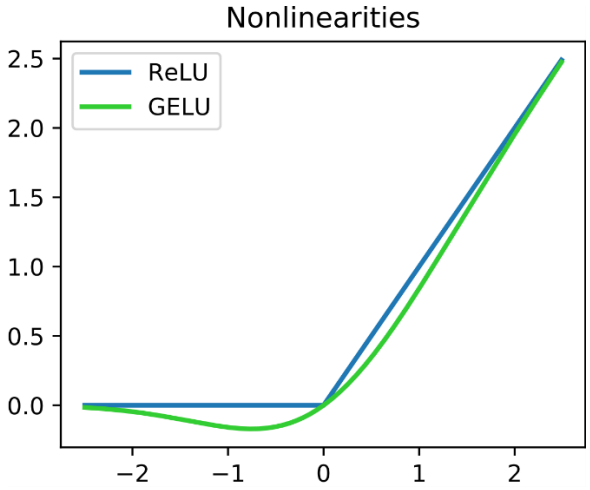

Figure 2.20: Comparison of ReLU and GELU non-linearity [35].

2.8.3 Softmax Activation Function

The output of a preceding layer is converted into a matrix of probabilities using the Softmax activation function. The classification of many classes frequently uses it.

$$P(y = j \mid x\} = \frac{e^{x^T w_j}}{\sum_{k=1}^{K} e^{x^T wk}}$$

2.9 Optimizers

2.9.1 Adam

Adam [17] is a scaling and momentum-based adaptive learning rate optimization technique that combines the advantages of RMSProp optimizer and SGD optimizer without Momentum. The optimizer is intended to be adaptable to non-stationary targets and issues

with gradients that are extremely noisy and/or sparse. The weight updates take place as follows:

$$w_t = w_{t-1} - \eta \frac{\widehat{m_t}}{\sqrt{\widehat{v_t}} + \epsilon}$$

With

$$\widehat{m_t} = \frac{m_t}{1 - \beta_1^t}$$

$$\widehat{v_t} = \frac{v_t}{1 - \beta_2^t}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

The step size or learning rate is "η" which in the original paper was about 1e-3. A tiny number, usually 1e-8 or 1e-10, is used as the separator to avoid dividing by 0.

2.9.2 AdamW

By separating weight decay from the gradient update, AdamW [18], a stochastic optimization technique, alters the way weight decay is typically implemented in Adam. To illustrate this, consider how L2 regularization in another adaptive optimizer such as in Adam is often handled, where W_t is the degree of weight decay during period t:

$$g_t = \nabla f(\theta_t) + w_t \theta_t$$

AdamW modifies the weight decay element so that it will show in the gradient change while:

$$\theta_{t+1,i} = \theta_{t,i} - \eta \left(\frac{1}{\sqrt{\widehat{v_t} + \epsilon}} \cdot \widehat{m_t} + w_{t,i} \theta_{t,i} \right), \forall t$$

2.10 Loss Functions

One of the most crucial components of neural networks is the loss function, which, together with the optimization functions, is directly in charge of fitting the model to the provided training data. To reduce algorithmic error, neural networks employ optimizing techniques such stochastic gradient descent. We really use a Loss Function to calculate this deviation. It's employed to express how well or poorly the model is working. These are separated into the two categories of Classification Loss and Regression Loss. In order to determine how well a neural network represents the training data, the loss function is used to compare the target and anticipated output values. We try to keep this difference in outputs between expected and goal as little as possible throughout training.

The hyperparameters are modified to reduce the average loss; we identify optimal weight, w^T, and bias, b, that reduce the value of J.

$$J(w^T, b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$$

2.10.1 Loss Function Types

Regression, as well as classification loss functions, are the two primary categories of loss functions in supervised learning, which correspond to the two primary categories of neural networks.

1. Regression Loss Functions are employed in regression neural nets; instead of using pre-selected labels, the model predicts an output value matching an input value when given that input value. For instance, Mean Squared Error(MSE) and Mean Absolute Error(MAE).
2. Classification loss functions are used in classification neural nets. Considering an input, the neural network generates a matrix of probability that the input belongs to several pre-set classes. The neural net can then choose the class with the highest likelihood of matching. Ex. Binary Cross-Entropy(BCE) Loss, Categorical Cross-Entropy(CCE) Loss.

2.10.2 Mean Squared Error

The MSE loss function, which is among the most well-liked ones, computes the norm of the squared deviations between the target and projected outputs. This algorithm is particularly well suited for computing loss because of a number of its characteristics. Since the distance is squared, it makes no difference if the projected value is greater or lower than the desired value; nevertheless, values with a significant inaccuracy are punished.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2$$

The fact that MSE is a convex function with a distinct global minimum (as depicted in the picture above) makes it easier for us to use a gradient descent algorithm to determine the weight values.

2.10.3 Mean Absolute Error

By averaging the absolute disparities between both the intended and anticipated outputs, MAE determines the target output. In some circumstances, this loss function is employed in place of MSE. Since the loss is doubled, the MSE is extremely prone to outliers, which may have a significant impact.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y^{(i)} - \hat{y}^{(i)}|$$

To counteract this, MAE is utilized when the training set contains a significant number of outliers.

2.10.4 Binary Cross-Entropy Loss

In order to determine which of the predetermined categories the provided input will fall under, classification neural nets first output a matrix of probabilities, and then they choose the group with the highest likelihood as the desired outcome.

$$CE\ Loss = \frac{1}{n}\sum_{i=1}^{N} -\ (y_i \cdot log(p_i) + (1 - y_i) \cdot log(1 - p_i)$$

There are only two real values of y that can be classified as valid in binary classification: 0 or 1. It is necessary to analogize the exact value (0 or 1) with the likelihood that the input falls into that category (P(i) = likelihood that the group is 1; 1 - P(i) = likelihood that the group is 0) in order to appropriately calculate the loss in between real and expected values.

2.10.5 Categorical Cross-Entropy Loss

When there are more than two different classes, we use categorical cross-entropy, which works in a manner quite akin to binary cross-entropy.

$$CE\ Loss = -\frac{1}{n}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} \cdot log(p_{ij})$$

Categorical cross-entropy has a specific instance known as binary cross-entropy, whereby M = 2 denotes the quantity of classes.

Chapter 3

Results & Discussion

The experimental findings of our study, which were obtained after classifying the dataset using the methods we suggested, will be discussed in this part. Additionally, a thorough explanation of the experimental design and steps used to arrive at the results will be provided. To assess the outcomes, we made use of the TensorFlow [19] and scikit-learn [20] libraries. The output is visualized using the Matplotlib [21] Library. The key concepts for analyzing our findings are covered first.

3.1 Classification Accuracy

The percentage of accurate predictions produced by a system out of all possible predictions in classification tasks. When we utilize the term accuracy, we typically imply classification accuracy. It measures the proportion of accurate predictions to all input samples. Only when there are an equivalent number of specimens from each class does it function properly. Consider, for instance, that in our training set, 98% of the samples are from class A and 2% are from class B. Then, by simply estimating that every training data belongs to class A, the model can effortlessly achieve 98% training accuracy.

$$Accuracy = \frac{True_{positive} + True_{negative}}{True_{positive} + True_{negative} + False_{positive} + False_{negative}}$$

The test accuracy will indeed decrease to 60% if the same system was evaluated on a testing dataset with 60% observations from class A and 40% observations from class B. Although classification accuracy is excellent, it gives us the impression that we have achieved high accuracy. When the costs of misclassifying the minor category instances are very large, a serious issue arises. The expense of missing diagnosing an individual with an unusual but debilitating disease is far higher than the expense of subjecting a normal individual to additional tests. When there is a majority with one class in the target attribute categories in the data, accuracy should not be utilized as a measurement.

3.2 Precision

The accuracy metric's drawback is overridden by the precision metric. The fraction of positive predictions that were accurate is determined by precision. It can be measured as the True Positive, or the proportion of total positive estimates that come true (True Positive and False Positive). The accuracy is determined by the ratio of Samples tested that were correctly classified to all samples that were categorized as Positive. (either correctly or incorrectly). The precision gauges how well the model categorizes an instance as positive. The denominator rises and the precision becomes low when the system makes many wrong Positive classifications or very few correctly Identified classifications.

$$Precision = \frac{True_{positive}}{True_{positive} + False_{positive}}$$

Contrarily, the precision is increased when:

1. The model constructs a lot of accurate Positive categorizations (improving the True Positive).
2. The model categorizes more Positives correctly (lessening the False Positive).

How accurately the model classifies objects as Positive is indicated by the precision. Precision is to accurately categorize all Positive cases as Positive and avoid incorrectly classifying a negative example as Positive. Identifying all Positive cases as Positive and correctly characterizing no Negative samples as Positive are both necessary for achieving 100% accuracy.

3.3 Recall

The true-positive rate (TPR), which refers to the proportion of true positives to all positives, is provided by the recall. The recall is determined as the proportion of Positive instances that were managed to identify as Positive to all Positive samples. The recall reflects how well the model can identify samples that were positive. The far more positive items that are identified, the larger the recall. Only the classification of the samples tested is important to the recall. Regardless of the way the negative observations are categorized, such as for the precision, this still holds true. Regardless of whether all of the negative instances were mistakenly classified as Positive, the recall will indeed be 100% whenever the model labels all of the samples tested as Positive.

$$Recall = \frac{True_{positive}}{True_{positive} + False_{negative}}$$

The classifier that categorizes all this as A can have a recall score of 0% for the positive category, B (precision would just be undefined — 0/0), in the scenario when classes A and B are split 99/1. When there is a class imbalance, precision and recall offer a more useful way to assess the performance of the model.

3.4 F1 Score

An F1 score's best value is 1 (perfect precision and recall), while its lowest value is 0. The F1 score is the harmonic norm of recall and precision. That's because the harmonic mean of a set of numbers strongly favors the list's lowest scores, it has the tendency (in contrast to the arithmetic average) to lessen the influence of extreme examples while amplifying the effect of minor ones. An integer in the middle of 0 and 1 will represent the F1 score. The F1 score of 1.0 denotes faultless recall and precision. If somehow the F1 score is 0, either the recall or the precision is also 0.

$$F1\ score = \frac{2*Precision*Recall}{Precision+Recall}$$

A higher F1 score penalizes excessive values. In the best-case scenario, an F1 Score could be a useful evaluation statistic in the classification scenarios listed below:

1. Adding extra data doesn't significantly affect the outcome provided False Positives (FP) and False Negatives (FN) are equivalently expensive, i.e., they miss true positives or uncover false positives in roughly equal amounts.
2. True Negative is elevated.

3.5 Confusion Matrix

A very helpful tool for determining the model's flaws (or strengths) is a confusion matrix. It is a matrix that contrasts the proportion of accurate and inaccurate predictions for each category.

There are four numbers to pay heed to in a confusion matrix.

1. True positives: How many data were positive that the model properly identified.

2. False-positive rate: The percentage of negative observations that the model mistakenly thought were positive.
3. True negative: The number of negative occurrences accurately predicted as negative by the model.
4. False-negative rate: The proportion of positive occurrences that the model misinterpreted as negative.

3.6 Experimental Setup

Without any pretraining or supplementary data, we assess our suggested model largely using the CIFAR10 [10] classification. Typical data augmentation methods like RandAugment [12], AutoAugment [11], MixUp [22], and CutMix [13] were employed. We employed an exponential learning rate scheduler [23] and the AdamW [18] optimizer. The learning rate for the AdamW optimizer started out at 1e-5 and rose to 0.01 after 5 warm-up epochs. After that, the learning rate dropped gradually to 5e-5. We used a predetermined number of epochs for our experiments. Our models were trained over 300 epochs. We just did a very small amount of hyperparameter tuning due to restricted computing resources. As a result, our models might be either over or under-fitted, and the accuracy we claim probably understates our model's potential. We executed all the experimentations on the Kaggle notebook with a TPU v3 hardware accelerator.

3.7 Model Specifications

We propose three different convolutional models based on the same architecture. The patch size, the kernel size of the depthwise separable layer, the projection dimension of the patches, and the depth of the model are differentiators among these models. We call these there versions Small, Medium, and Large based on the training parameters of these models. Although each type of model has a different patch projection dimension, the ratio of channel reduction before each Depthwise separable convolution was kept the same. Each of the model has 1*1 Convolution layer for shrinking the size of the image before depthwise convolution which resulted in fewer multiplicative and memory-intensive parameters. This also reduces the FLOPS of the model.

Table 3.1: Model Specifications.

Model Size	Patch Size	Projection Dim	Depth Conv. Kernel Size	Model Depth	Trainable Parameters	Flops
Small	2	128	5	8	130,442	0.0325G
Medium	2	256	5	8	490,250	0.1238G
Large	1	256	7	16	998,666	1.0092G

3.8 Experimental Results on the CIFAR10 Dataset

3.8.1 Classification Report

Table 3.2: Classification Report of Small Model.

Class	Precision	Recall	F1-score
0	0.93	0.94	0.93
1	0.96	0.98	0.97
2	0.93	0.91	0.92
3	0.88	0.82	0.85
4	0.92	0.94	0.93
5	0.87	0.88	0.88
6	0.93	0.97	0.95
7	0.95	0.95	0.95
8	0.95	0.96	0.96
9	0.98	0.95	0.96
Macro Avg	0.93	0.93	0.93
Weighted avg	0.93	0.93	0.93

Table 3.3: Classification Report of Medium Model.

Class	Precision	Recall	F1-score
0	0.97	0.97	0.97
1	0.97	0.98	0.98
2	0.95	0.94	0.95
3	0.91	0.87	0.89
4	0.95	0.96	0.95
5	0.92	0.92	0.92
6	0.95	0.99	0.97
7	0.98	0.97	0.97
8	0.97	0.98	0.98
9	0.97	0.96	0.97
Macro Avg	0.95	0.95	0.95
Weighted avg	0.95	0.95	0.95

Table 3.4: Classification Report of Large Model.

Class	Precision	Recall	F1-score
0	0.98	0.97	0.98
1	0.98	0.98	0.98
2	0.97	0.97	0.97
3	0.94	0.91	0.92
4	0.96	0.98	0.97
5	0.91	0.94	0.94
6	0.98	0.99	0.98
7	0.99	0.97	0.98
8	0.98	0.99	0.98
9	0.98	0.97	0.98
Macro Avg	0.97	0.97	0.97
Weighted avg	0.97	0.97	0.97

3.8.2 Classification Accuracy

Table 3.5: Accuracy on the CIFAR10 dataset.

Model Size	Trainable Parameters	Training Accuracy	Testing Accuracy	Epochs	FLOPS
Small	130,442	71%	93%	300	0.0325G
Medium	490,250	76%	95%	300	0.1238G
Large	998,666	82%	97%	300	1.0092G

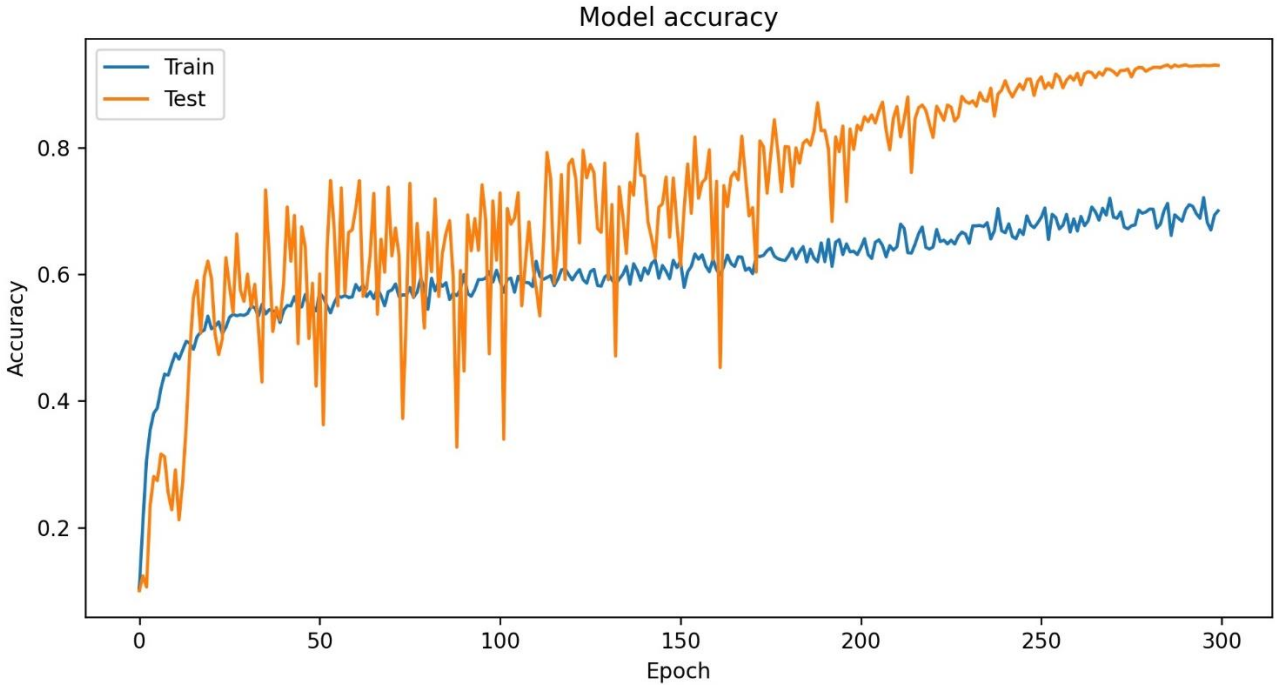

Fig 3.1: Accuracy of the Proposed Small Model.

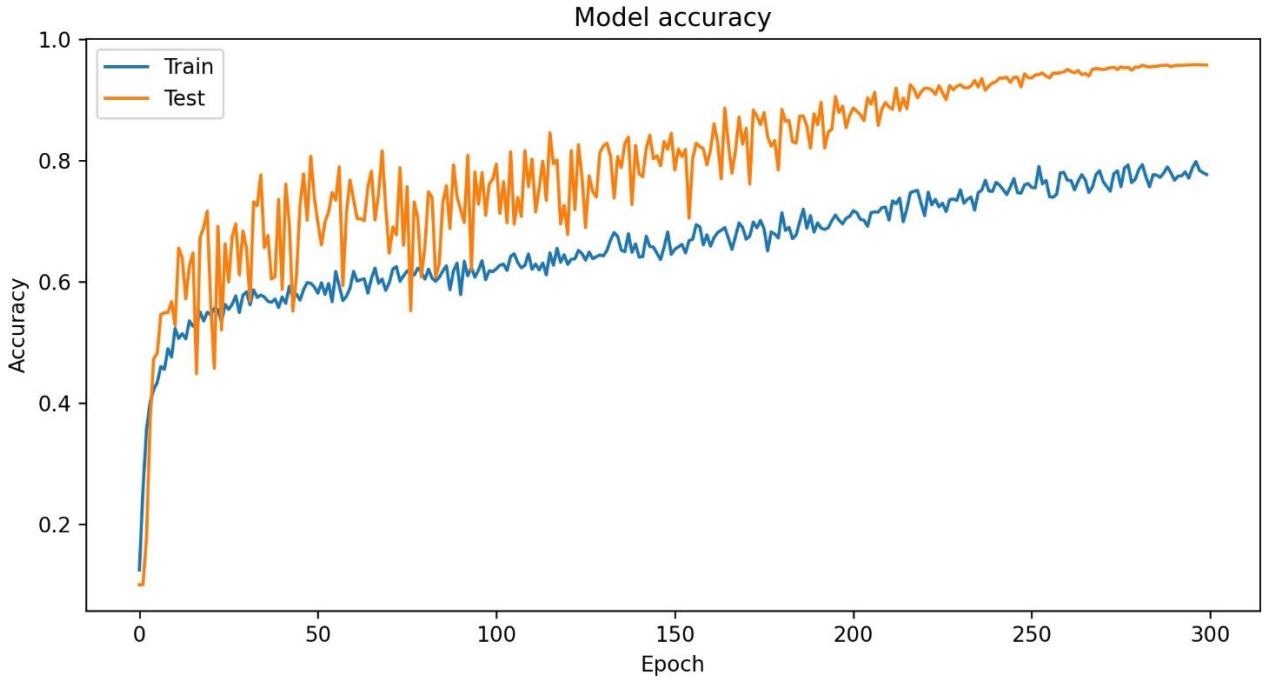

Fig 3.2: Accuracy of the Proposed Medium Model.

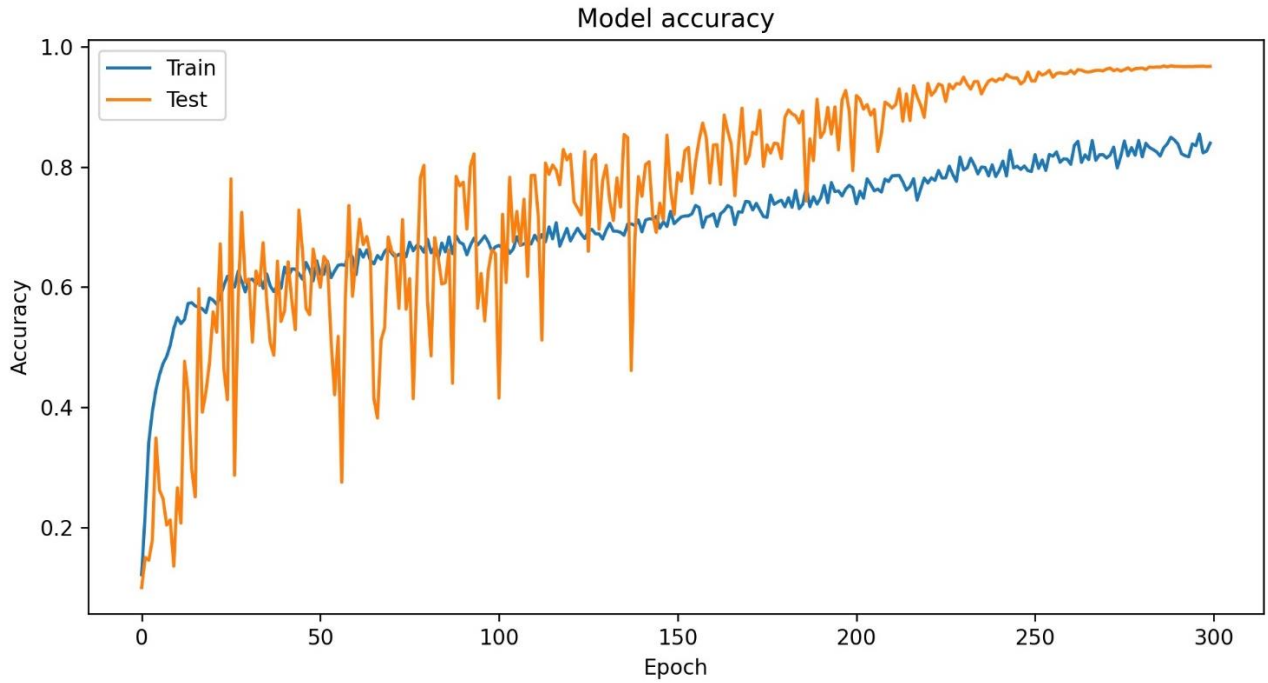

Fig 3.3: Accuracy of the Proposed Large Model.

3.8.3 Confusion Matrix

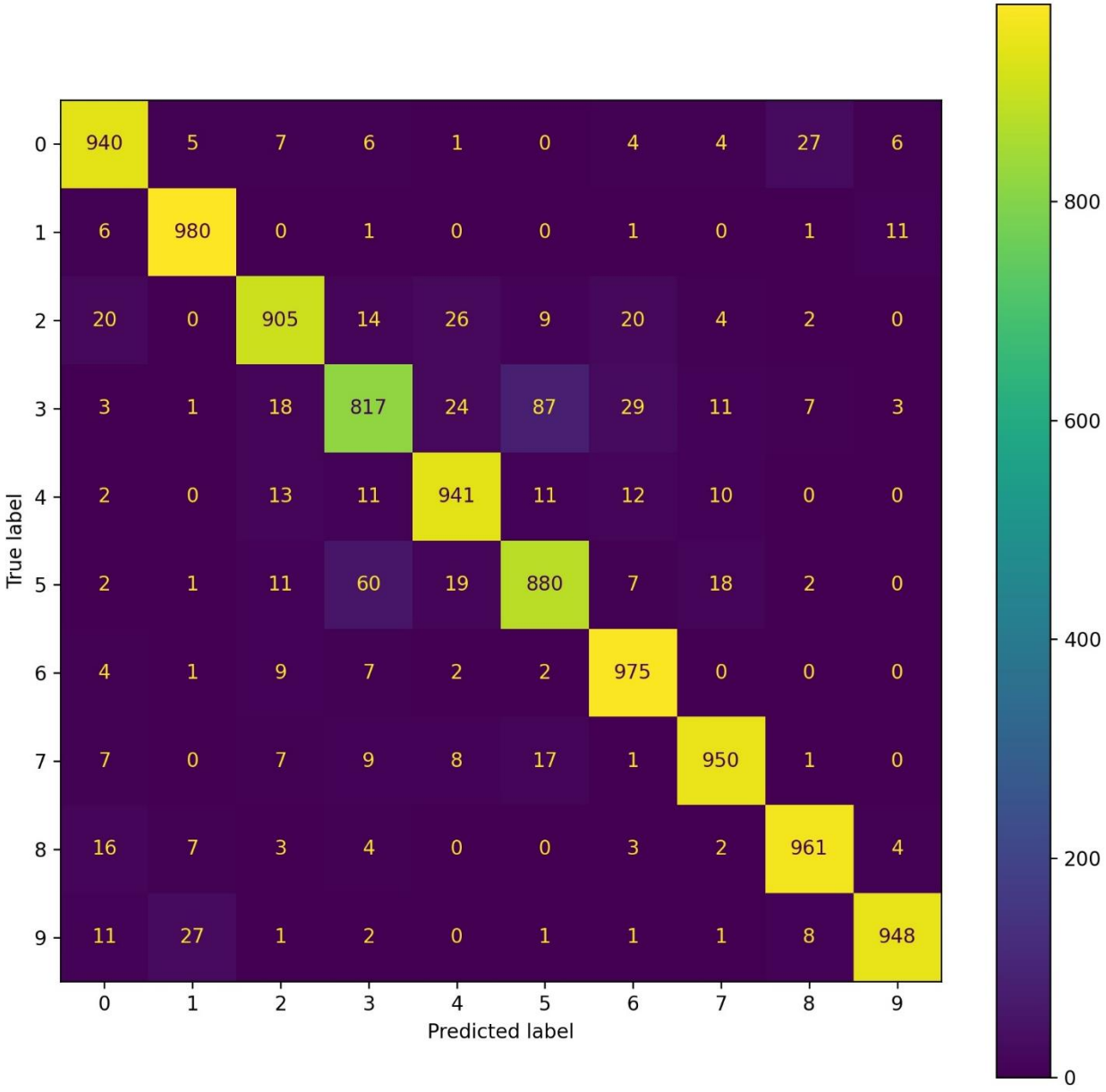

Fig 3.4: Confusion Matrix of the Proposed Small Model.

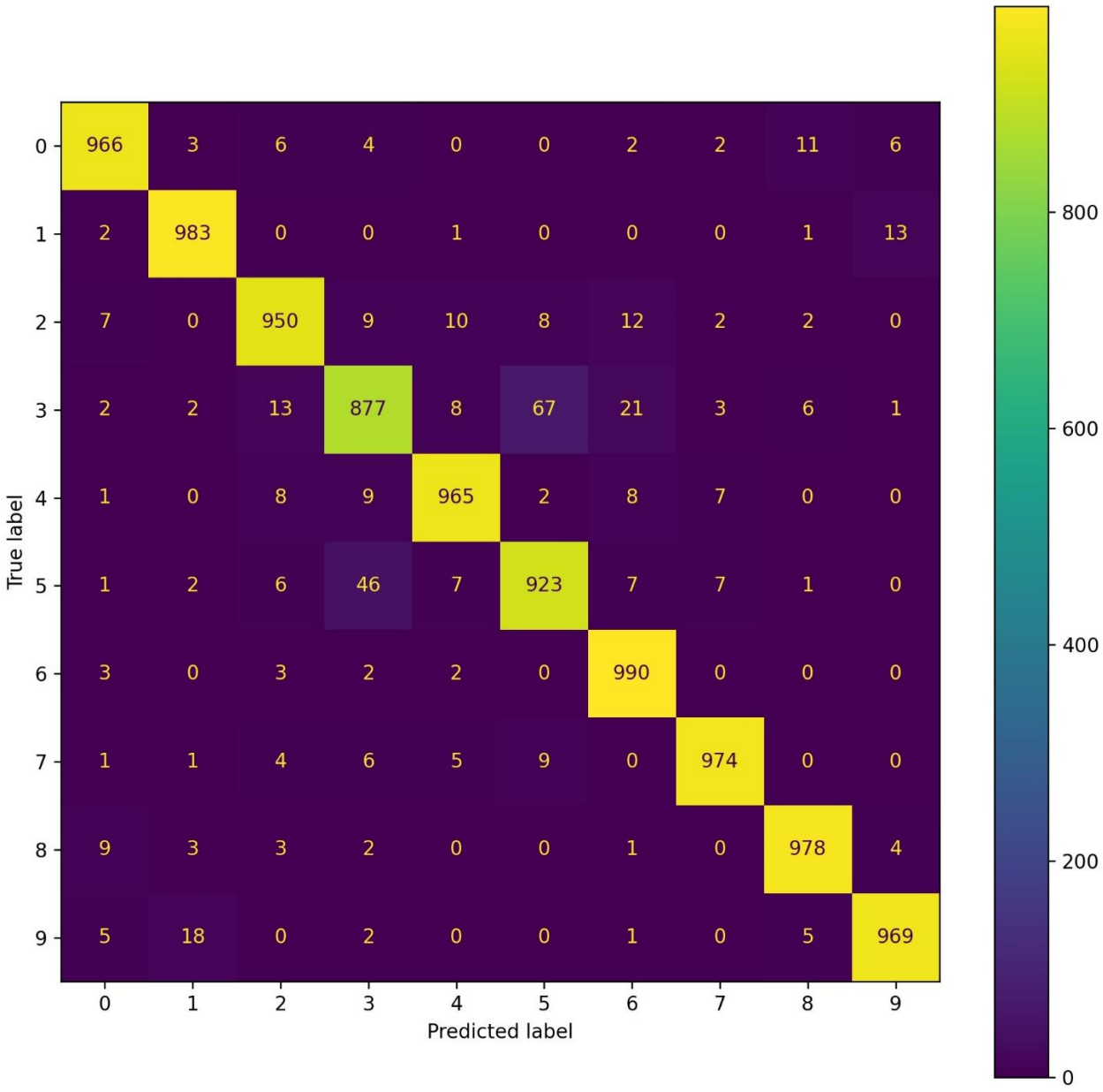

Fig 3.5: Confusion Matrix of the Proposed Medium Model.

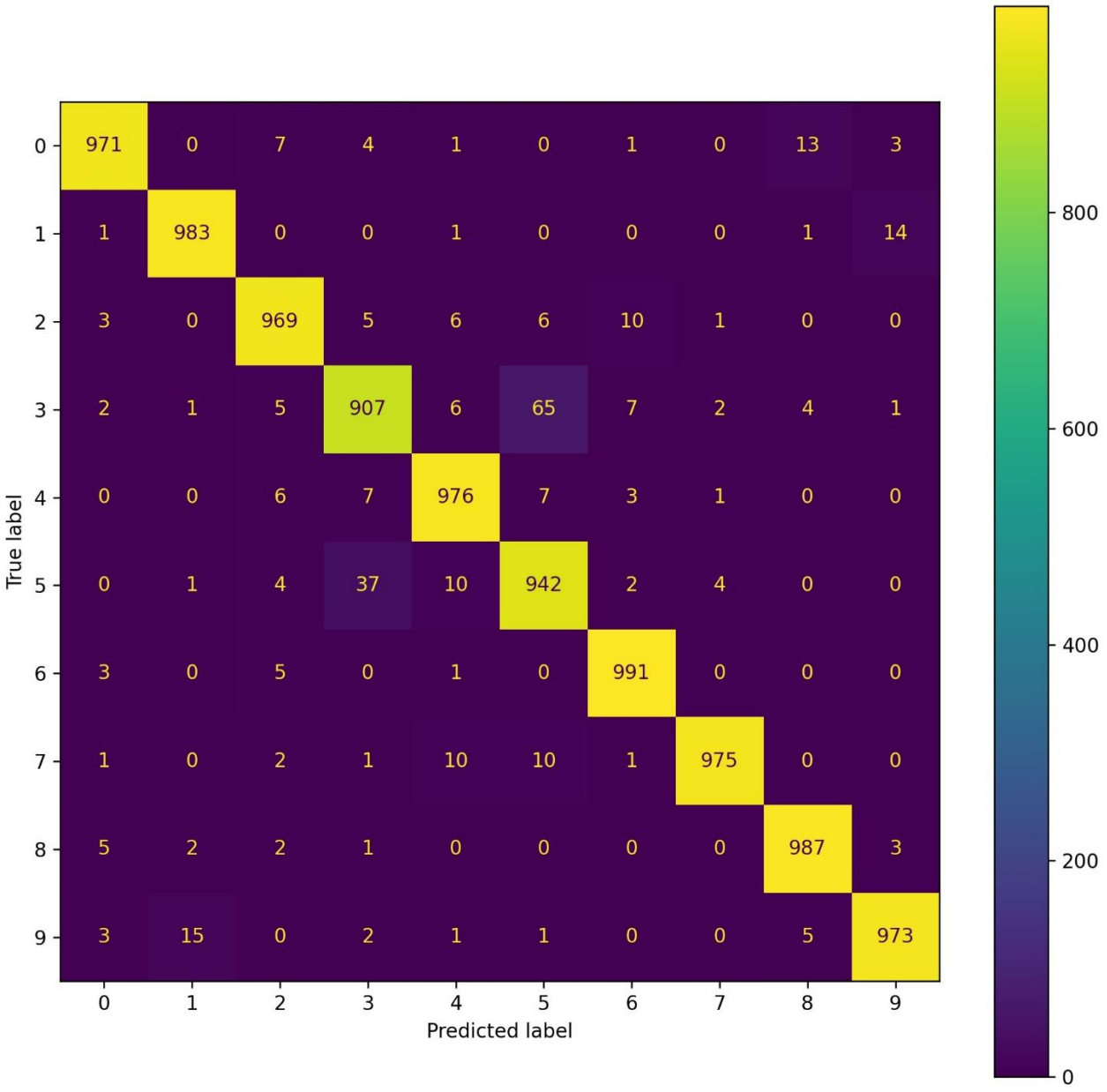

Fig 3.6: Confusion Matrix of the Proposed Large Model.

3.9 Discussion

This chapter has covered our research on CIFAR10 image classification based on our proposed image classifier. Our three suggested models, which are based on the same architecture, have been tested to see how well they perform. We have observed that the model's accuracy increases with an increase in the depthwise separable convolution kernel size. However, excessive kernel sizing has negative effects on model parameters, FLOPS, and training time. In our experiment, shrinking the size of the image before depthwise convolution resulted in fewer multiplicative and memory-intensive parameters. We have also observed the significance of patch size on the model's overall classification accuracy. We have found that the accuracy of these models is highest when a patch size is as small as 1. The model's performance was also significantly influenced by the depth of the model. Although adding depth increases complexity, the improvement in accuracy shows that the added depth is not likely to cause the model to overfit. The model was not at all overfitted because of the extensive use of augmentation techniques that served as regulation techniques; instead, testing accuracy was higher than training accuracy. This demonstrates that our model is more broadly generalizing the training data set. So, the proposed model might be ideal in the case where the training data amount is small, and the computational budget is limited.

Chapter 4

Conclusion & Future Work

4.1 Conclusion

In this thesis, we applied supervised learning to the problem of image classification. The primary goal of this research was to develop a new image classification architecture with fewer parameters and complexity. By selecting appropriate data for testing our model, experimenting with the best optimizer, weight decay and learning rate of the optimizer, using appropriate regularization techniques, and more, we have taken a methodical approach to achieve our goal. Three classification models that are based on the same architecture as the one we proposed are presented here. In comparison to other larger CNN or Vision Transformer [4] models, our models may not achieve state-of-the-art accuracy, but the experimental data demonstrates a good speed-accuracy trade-off.

A two-stage 1*1 convolutional layer, the main building block of our network, reduces the input channels for depthwise convolution. We can achieve fewer parameters and a shorter training time thanks to this reduction in dimensionality. We have employed pointwise convolution in an unconventional manner. The pointwise convolution typically employs the same number of filters as the depthwise convolution of the preceding layer. However, we suggested using pointwise convolution to expand the input channel's dimension. By doing so, we are able to maintain a residual connection to the input that came before the dimension was reduced. As a result, the flow of information remains constant. This also solved the vanishing gradient and exploding gradient problems.

We have observed a 3-7% increase in accuracy when using Batch Normalization [14], despite Layer Normalization [24] becoming more and more popular in Transformer, MLP, and even CNN models. The use of the AdamW [18] optimizer instead of the Adam [17] optimizer resulted in a 2-3% increase in accuracy. The regularization techniques used aided in preventing overfitting. The findings indicate that none of the models were overfitted. One of the most significant key findings of our experiment is that the patch-based image

representation method introduced in Vision Transformers also aids the performance of the CNNs.

Although positional encoding is necessary for Vision Transformers [4], our model did not require it because our architecture did not call for self-attention. Accuracy is improved by Convolution's inductive bias over the image patches. Another finding of our experiment was the use of Batch Size. The accuracy was lower when we used a batch size of 128 than it was when we used a batch size of 512, despite the fact that theoretically, the smaller batch size should increase accuracy. As a result, we kept the batch size at 512 for all of our tests. This might be the result of using Cloud TPUs for our training rather than GPUs.

As the use of AI in our daily lives grows, so does the demand for compact image classifiers. The internet has made data shortages less frequent, but there are still some edge cases where we need an effective image classification model with a good complexity-accuracy tradeoff. We are witnessing a rapid rise in computer vision in many industries, including the medical sector, satellite imaging, agriculture, and medicine, but the amount of data available in these industries is insufficient. We anticipate that our suggested architecture will serve as the foundation for many computer vision tasks in these fields, including segmentation, object detection, and classification.

4.2 Future Work

The CIFAR10 [10] dataset, which has images with a size of 32*32, served as the basis for the entire research process. High-resolution image data was not available for us to test our model on. This resulted from the fact that we had limited computational resources. Additionally, we limited the testing of our suggested model to image classification tasks. Therefore, testing and evaluating our model on large, high-resolution image datasets like ImageNet would make up the majority of our study's future work. We did not evaluate the effectiveness of our model in other computer vision fields. Future research can also be extended to evaluate how well our model performs on other computer vision tasks like object detection and image segmentation. The efficiency of the model as a whole could have been increased by further hyperparameter tuning. To put our theoretical research into practice, we want to implement our classification model on edge devices like smartphones for real-time image classification.

References

[1] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv [cs.CV], 2014.

[2] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proc. IEEE Inst. Electr. Electron. Eng., vol. 86, no. 11, pp. 2278–2324, 1998.

[3] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” arXiv [cs.CV], 2015.

[4] A. Dosovitskiy et al., “An image is worth 16x16 words: Transformers for image recognition at scale,” arXiv [cs.CV], 2020.

[5] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1800–1807.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” Commun. ACM, vol. 60, no. 6, pp. 84–90, 2017.

[7] A. Vaswani et al., “Attention is all you need,” arXiv [cs.CL], 2017.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248–255.

[9] C. Szegedy et al., “Going deeper with convolutions,” arXiv [cs.CV], 2014.

[10] “CIFAR-10 and CIFAR-100 datasets,” Toronto.edu. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html. [Accessed: 15-Oct-2022].

[11] E. D. Cubuk, B. Zoph, Mane, Dandelion, V. Vasudevan, and Q. V. Le, “AutoAugment: Learning augmentation policies from data,” arXiv [cs.CV], 2018.

[12] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, “RandAugment: Practical automated data augmentation with a reduced search space,” arXiv [cs.CV], 2019.

[13] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, “CutMix: Regularization strategy to train strong classifiers with localizable features,” arXiv [cs.CV], 2019.

[14] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating deep network training by reducing internal covariate shift,” arXiv [cs.LG], 2015.

[15] D. Hendrycks and K. Gimpel, “Gaussian Error Linear Units (GELUs),” arXiv [cs.LG], 2016.
[16] A. F. Agarap, “Deep Learning using Rectified Linear Units (ReLU),” arXiv [cs.NE], 2018.
[17] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” arXiv [cs.LG], 2014.
[18] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” arXiv [cs.LG], 2017.
[19] T. Developers, TensorFlow. Zenodo, 2022.
[20] F. Pedregosa et al., “Scikit-learn: Machine Learning in Python,” arXiv [cs.LG], no. 85, pp. 2825–2830, 2012.
[21] J. D. Hunter, “Matplotlib: A 2D Graphics Environment,” Comput. Sci. Eng., vol. 9, no. 3, pp. 90–95, 2007.
[22] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “mixup: Beyond Empirical Risk Minimization,” arXiv [cs.LG], 2017.
[23] Z. Li and S. Arora, “An exponential learning rate schedule for deep learning,” arXiv [cs.LG], 2019.
[24] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer Normalization,” arXiv [stat.ML], 2016.
[25] “Image Classification,” Huggingface.co. [Online]. Available: https://huggingface.co/tasks/image-classification. [Accessed: 15-Oct-2022].
[26] “Supervised Machine learning - javatpoint,” www.javatpoint.com. [Online]. Available: https://www.javatpoint.com/supervised-machine-learning. [Accessed: 15-Oct-2022].
[27] “CutMix data augmentation for image classification,” Keras.io. [Online]. Available: https://keras.io/examples/vision/cutmix/. [Accessed: 15-Oct-2022].
[28] D. Shrinet, “Image based search engine with CNN and Transfer Learning,” The Startup, 15-Jun-2020. [Online]. Available: https://medium.com/swlh/image-based-search-engine-with-cnn-and-transfer-learning-153a1a3e58b4. [Accessed: 15-Oct-2022].
[29] IBM Cloud Education, “What are Convolutional Neural Networks?,” Ibm.com, 20-Oct-2020. [Online]. Available: https://www.ibm.com/cloud/learn/convolutional-neural-networks. [Accessed: 15-Oct-2022].

[30] “CNN,” GeeksforGeeks, 05-Aug-2019. [Online]. Available: https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/. [Accessed: 15-Oct-2022].
[31] “Overview of convolutional neural networks,” Topcoder.com. [Online]. Available: https://www.topcoder.com/thrive/articles/overview-of-convolutional-neural-networks. [Accessed: 15-Oct-2022].
[32] “Depthwise Separable Convolution,” Paperswithcode.com. [Online]. Available: https://paperswithcode.com/method/depthwise-separable-convolution. [Accessed: 15-Oct-2022].
[33] A. Pandey, “Depth-wise convolution and depth-wise separable convolution,” Medium, 09-Sep-2018. [Online]. Available: https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec. [Accessed: 15-Oct-2022].
[34] “Papers with code - ReLU explained,” Paperswithcode.com. [Online]. Available: https://paperswithcode.com/method/relu. [Accessed: 15-Oct-2022].
[35] Wikipedia contributors, “Rectifier (neural networks),” Wikipedia, The Free Encyclopedia, 19-Sep-2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Rectifier_(neural_networks)&oldid=1111203702.