Bachelor of Science in Computer Science
May 2021

A Deep Learning Application for Traffic Sign Classification

Pramod Sai Kondamari
Anudeep Itha

Faculty of Computing, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

Contact Information:
Author(s):
Pramod Sai Kondamari
E-mail: prko20@student.bth.se

Anudeep Itha
E-mail: ania20@student.bth.se

University advisor:
Shahrooz Abghari
Department of Computer Science

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet	:	www.bth.se
Phone	:	+46 455 38 50 00
Fax	:	+46 455 38 50 57

Abstract

Background: Traffic Sign Recognition (TSR) is particularly useful for novice drivers and self-driving cars. Driver Assistance Systems(DAS) involves automatic traffic sign recognition. Efficient classification of the traffic signs is required in DAS and unmanned vehicles for safe navigation. Convolutional Neural Networks(CNN) is known for establishing promising results in the field of image classification, which inspired us to employ this technique in our thesis. Computer vision is a process that is used to understand the images and retrieve data from them. OpenCV is a Python library used to detect traffic sign images in real-time.
Objectives: This study deals with an experiment to build a CNN model which can classify the traffic signs in real-time effectively using OpenCV. The model is built with low computational cost. The study also includes an experiment where various combinations of parameters are tuned to improve the model's performance.
Methods: The experimentation method involve building a CNN model based on modified LeNet architecture with four convolutional layers, two max-pooling layers and two dense layers. The model is trained and tested with the German Traffic Sign Recognition Benchmark (GTSRB) dataset. Parameter tuning with different combinations of learning rate and epochs is done to improve the model's performance. Later this model is used to classify the images introduced to the camera in real-time.
Results: The graphs depicting the accuracy and loss of the model before and after parameter tuning are presented. An experiment is done to classify the traffic sign image introduced to the camera by using the CNN model. High probability scores are achieved during the process which is presented.
Conclusions: The results show that the proposed model achieved 95% model accuracy with an optimum number of epochs, i.e., 30 and default optimum value of learning rate, i.e., 0.001. High probabilities, i.e., above 75%, were achieved when the model was tested using new real-time data.

Keywords: Image Processing, Deep Learning Algorithms, Convolutional Neural Network (CNN), OpenCV, Supervised Learning.

Acknowledgments

We are grateful to our supervisor, Shahrooz Abghari, PhD, for mentoring us throughout our thesis. We could not complete this assignment without the help of our family and friends. They sincerely believed in our work and helped us give ourselves time to do this thesis. We want to thank every person who inspired us for completing the thesis.

Authors:
Pramod Sai Kondamari
Anudeep Itha

Contents

Abstract **i**

Acknowledgments **iii**

1 Introduction **1**
1.1 Aim and Objectives 2
1.2 Research Question 2
1.3 Overview 3

2 Background **4**
2.1 Deep Learning 4
2.2 Supervised Learning 4
2.3 Neural Networks 5
2.4 Convolutional Neural Network 5
2.5 Activation Functions 6
2.6 Optimization Algorithm 7
2.7 Loss Function 8
2.8 Computer Vision 8
2.9 Image Processing 8
2.10 Image Augmentation 9
2.11 Performance Evaluation Metrics 9

3 Related Work **10**

4 Method **12**
4.1 Experimentation Environment 13
4.2 Data Overview 15
4.3 Processing The Dataset Images 16
4.4 Image Augmentation 16
4.5 Building The CNN Model 18
4.6 Training and Testing The Model 19
4.7 Parameter Tuning 20
4.8 Testing The Model In Real-time 20

5 Results and Analysis **22**
5.1 Experiment-1: Training The Model 22
5.1.1 The model trained with default parameter values 22

5.1.2 The model trained after parameter tuning with various combinations 23
5.2 Experiment-3: Using The Model In Real-time 30

6 Discussion **32**

7 Conclusions and Future Work **35**
7.1 Conclusion 35
7.2 Future work 35

Bibliography **36**

A Model code snippet **40**

List of Figures

2.1 In the figure a convolution filter (Sobel Gx) is applied on each sub-array of the source pixel to obtain a corresponding destination pixel (The figure is borrowed from [2]) 6

4.1 Steps involved in training, testing the proposed model (Phase 1) and testing the model in real-time (Phase 2). 13
4.2 Sample of the GTSRB dataset 15
4.3 Sample Pre-processed Image 17
4.4 Augmented Images 17
4.5 Our proposed CNN model Architecture 18
4.6 Distribution of the training dataset 20

5.1 Model accuracy with default parameter values 22
5.2 Loss occurred 23
5.3 Accuracy 24
5.4 Loss 24
5.5 Accuracy 25
5.6 Loss 25
5.7 Accuracy 26
5.8 Loss occurred 26
5.9 Accuracy 27
5.10 Loss occurred 27
5.11 Accuracy 28
5.12 Loss occurred 28
5.13 Accuracy 29
5.14 Loss occurred 29
5.15 Traffic sign classification in real time using webcam 30

6.1 Graph plot reflecting the model learning patterns with decreasing number of epochs. (The figure is borrowed from [3]) 33

List of Tables

4.1 Distribution of images across different datasets 15
4.2 Architecture of CNN 18
4.3 Various combinations of parameter tuning 20

5.1 Probability scores achieved when testing the model with new images in real time. 31

6.1 Accuracy and loss of model for train and test data by tuning various combinations of learning rate and epochs 32

Chapter 1

Introduction

Traffic signs are indicators present at either the side or above roads to provide information about the road conditions and directions. Traffic signs are generally pictorial signs using symbols and pictures to give information to the vehicle drivers. There are various categories of traffic signs classified according to the information they convey. There are primarily 7 different traffic signs [5]: 1) Warning signs, 2) Priority signs, 3) Restrictive signs, 4) Mandatory signs, 5) Special regulation signs, 6) Informative signs, 7) Directional signs and additional panels.

Among these signs, warning signs are most important. Warning signs indicate the potential hazard, obstacle, condition of the road. The warning signs may provide information about the state of the road or hazards on the road which the driver may not immediately see. This category includes pedestrian crossing sign, caution sign, crosswalk alert, traffic light alert etc. Priority signs indicate the course of the vehicle, i.e., the way a vehicle should pass to prevent a collision. This category includes stopping sign, intersection sign, one-way traffic sign etc. Restrictive signs are used to restrict certain types of manoeuvres and to prohibit certain vehicles on roads. This includes a no-entry sign, no-motorcycle sign, no-pedestrian etc. Mandatory signs are quite the opposite to prohibit signs saying the drivers what they must do. Permitted directions, Vehicle allowances belong to this category. Special regulation and informative signs indicate regulation or provide information about using something. One way sign, home zone indication comes to this category. Directional signs provide information about the possible destinations from a location. Turn-right, Turn-left etc., come under this category.

In recent years, several attempts are being made to detect the traffic signs and inform the vehicle driver. Traffic Sign Recognition regulates the traffic signs and informs the vehicle driver about the detected sign for efficient navigation, ensuring safety. A smart real-time automatic traffic sign detection can support the driver or, in the case of self-driving cars, can provide efficient navigation. Automatic traffic sign recognition has practical applications in Driver Assistance System [22]. DAS involves automatic traffic sign recognition where the traffic sign is detected and classified in real-time. Traffic sign recognition is also an important module in unmanned driving technology.

Traffic sign recognition systems usually have two phases [19, 32]. The first phase involves detecting the traffic sign in a video sequence or image using a method called image processing. The second phase involves classifying this detected image using an artificial neural network model. Detecting the traffic signs at various places and recognizing them has been quite an issue nowadays, which causes various problems.

According to World Health Organization (WHO), enforcing speed limits on traffic help minimizing road accidents [29]. This speed limit is often neglected by drivers who drive past the limits. Road accidents can be caused due to various reasons like the low quality of the traffic sign, driver negligence, obstacles near the traffic sign, which causes visual hindrance. The mentioned causes involve human-caused errors. Automated traffic sign detection may minimise human-involved errors and can effectively detect the traffic signs.

In recent years, deep learning have proven effective in many fields like speech recognition, natural language processing, image processing [17]. In particular, deep learning performed well and achieved better results in visual recognition tasks than humans [18].

In this thesis, we propose a cost-sensitive CNN model in computational resource cost like convolution layers. Our model is based on a modified LeNET model with four convolution layers that can classify the traffic signs. GTSRB dataset [4] is used to train and test the model. Later, we will use this model to classify the image introduced to the model via a camera. Here we use OpenCV to detect and extract the traffic sign picture placed in front of the camera.

1.1 Aim and Objectives

This thesis aims to build a cost-sensitive CNN model trained and tested using the GTSRB dataset. Later the model is used to classify the images placed in front of a camera detected using OpenCV.

Objectives:

- The first objective is to split the dataset, which contains images, into train and test data.
- To pre-process the images in the dataset using grayscale and histogram equalization methods.
- To build a cost-sensitive CNN model with a modified LeNet model.
- To train and test the model with the corresponding datasets.
- To test the hypothesis of whether the accuracy increases with parameter tuning and draw a suitable conclusion.
- To test the model with the images detected using OpenCV.

1.2 Research Question

RQ1:

Considering the initial parameter setups of the proposed model, to what extent can parameter tuning improve the model's accuracy?

Motivation:

Parameter tuning is used to get the best accuracy from the optimal combination of parameters by checking with all parameter combinations. Parameter tuning needs to be done to get better accuracy and minimum loss. We can achieve an optimum model using parameter tuning.

RQ2:

How can a traffic sign image be detected and extracted effectively when introduced to the live camera using OpenCV?

Motivation:

The built model has to be tested in real-time. We send an image extracted from the live camera using OpenCV to input the model for efficient classification. Data pre-processing needs to be done to recognize the image effectively. Grayscale and histogram equalization techniques are suitable for the given dataset.

1.3 Overview

This section discusses the structure of our thesis. The background contains various concepts that we have used in our thesis. This chapter contains a brief explanation for each choice in our work. Related work discusses previous works in the field of neural networks and computer vision for traffic sign recognition. The method comprises the procedure of our project, starting from the experimentation environment to the evaluation metrics used for evaluating the model. The results obtained from the method are explained in detail in the results chapter. Discussion about the obtained result is discussed in the discussion chapter. Finally, the thesis ends with a conclusion where deductions are made based on the results obtained, and future works are briefly discussed.

Chapter 2 Background

This chapter provides a brief explanation of the theoretical background necessary to understand the project report. In this section, we discuss various concepts and tools that are used in our thesis.

2.1 Deep Learning

Deep learning is the subfield of artificial intelligence. It deals with creating neural network models that can make data-driven decisions by themselves [15]. Deep learning enables systems to learn from experience and understand the world in terms of the hierarchy of concepts [24]. There is no human intervention to specify the knowledge needed for the computer as the computer is self-capable in gathering the information. Feature extraction is one of the characteristics that make deep learning special. Feature extraction is a concept where the properties are extracted from the input data provided to the model. In this way, image recognition can be achieved without any explicit coding.

Every image comprises pixels. These pixels are correlated to the neighbouring pixels. Deep learning techniques exploit this phenomenon by extracting features from small sections of a larger image. Such methods have the ability to learn high-level features so that the task of developing a feature extractor for every new problem is eliminated. As deep learning algorithm has many parameters than machine learning algorithms, it usually takes more time to train a deep learning model.

2.2 Supervised Learning

Supervised learning is one of the common branches of machine learning. Supervised learning requires learning a mapping between input and output data used to predict the outputs from unseen data [13]. In supervised learning, the model is trained with the labelled data, i.e., images with class labels in image classification. During the training phase, features are extracted from the training data and identify the pattern of the dataset. Test data is sent as an input to the model during the testing phase, where the data is classified according to the knowledge gained during the training phase. The main aim of supervised learning is to accurately predict the classification of the test data based on the prior knowledge gained by analyzing the train data. Artificial neural networks, Bayesian statistics, analytical learning are a few approaches and algorithms used in supervised learning [8].

2.3 Neural Networks

A neural network is an information processing system typically comprising biological neurons to implement learning and memorise things. When a neural network consists of software and hardware components to solve the Artificial Intelligence (AI) problems, they are called Artificial Neural Networks (ANN). A neural network comprising of artificial neurons is called an ANN. The network consists of several layers, including activation layers and algorithms. In this thesis, we use Convolutional Neural Network.

2.4 Convolutional Neural Network

Convolution networks (also known as CNN or ConvNet) belong to the family of neural networks, which can process data with a known grid-like topology [10]. In the case of images, there is a 2D grid of pixels. A CNN takes an input image and adds importance by adding weights and biases to various aspects/objects in the image and differentiating one image from another [7]. CNN simply employ a mathematical operation called convolution instead of matrix multiplication in its layers. CNN consists of an input layer and an output layer with multiple convolutional, pooling or fully connected layers in between [27]. The convolution operation can be mathematically expressed as:

$$s(t) = \int x(a)w(t-a)da \tag{2.1}$$

This can further be simplified to the following:

$$s(t) = (x * w)(t) \tag{2.2}$$

Here x is considered the input variable, i.e., the image pixels and w is considered the kernel or the filter applied to the pixels. A kernel is a matrix smaller in size than the image. This operation applies the kernel with the corresponding neighbourhood of image pixels to get an output called a feature map. The output of one layer is sent as an input to the other layer. In this way, the convolutions happen for the image pixels, and corresponding feature maps are generated from which the model extracts the features.

Convolutional layer

A CNN consists of several convolutional layers used to extract features from the input images. When the input image pixels passes through the convolutional layer, kernels are applied to the neighbourhood pixels of the image. Cross-correlation is performed on the kernel matrix and then applied to the image pixels. The weights in the kernel are adjusted during the training, but during calculations, the weights are constant. The Figure 2.1 shows the application of the kernel to the image pixels. After the convolution mapping, the output is sent to non-linear activation functions for handling the gradient problem. This is called feature mapping.

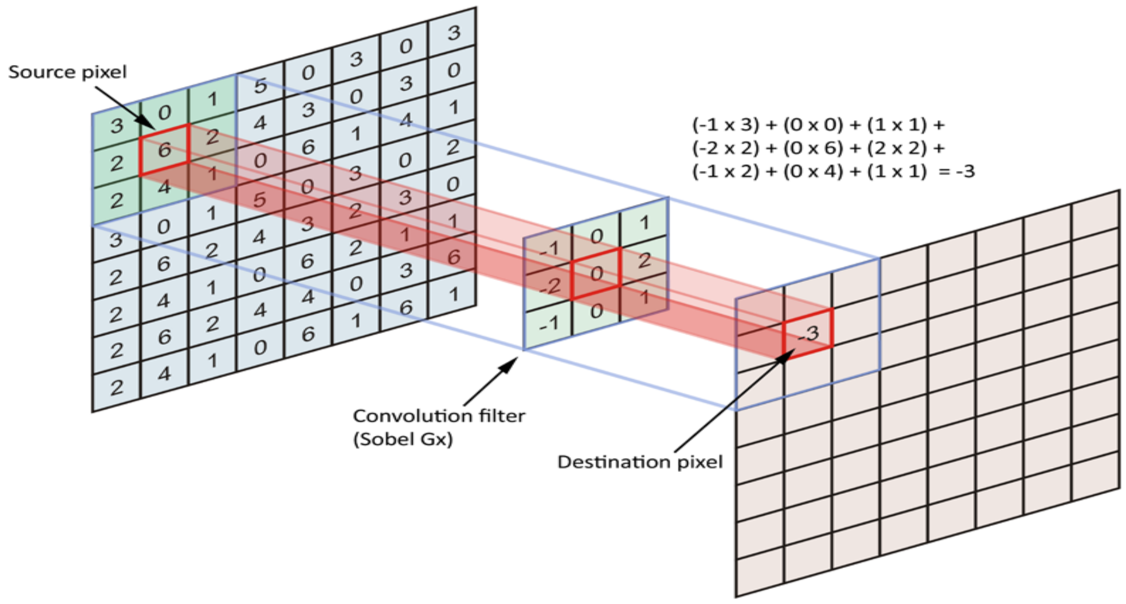
Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel

Figure 2.1: In the figure a convolution filter (Sobel Gx) is applied on each sub-array of the source pixel to obtain a corresponding destination pixel (The figure is borrowed from [2])

Pooling

Pooling is a sliding window technique where statistical functions are applied to the values within a window. Max pooling is a commonly used function where the maximum value within the window is taken. Pooling is used for downsampling operation. Pooling is mainly used to reduce the number of parameters and to make feature detection more robust.

Fully connected layer

A fully connected or dense layer makes the classification of the objects from the output obtained from the pooling layer. The input to the dense layer should be flattened into a single N x 1 tensor.

Dropout layer

The dropout layer reduces the overfitting problem by randomly setting the inputs to zero with configurable probability. The idea of dropout is to randomly drop units, along with their connections, during the training of the neural network. Dropout improves the performance of neural networks [35].

2.5 Activation Functions

Activation functions are used in ANN to convert the input data to output data, which is sent as input to the next layer. A neural network model's prediction accuracy is dependent on the type of activation function used [31]. If the activation functions are not used, the output data would be a simple linear polynomial function. The model will not be able to learn and recognize complex mappings from the data. Thus the neural network will not be effective for modelling complicated types of data such as images, speech, text, video, audio etc. ReLU and Softmax are two activation functions that we used in this thesis.

1. **ReLU**

   ReLU stands for rectified linear unit belongs to the non-linear activation function, which is widely used in neural networks. ReLU activation function has made training deep neural networks easier by reducing the burden of weight-initialization and vanishing gradients [6]. ReLU is more efficient than other activation functions as only a few neurons are activated at a time rather than activating all neurons. ReLU is a piecewise linear function whose positive part is an identity function (x), and the negative part is zero [28]. ReLU function can be mathematically represented as:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

2. **Softmax**

   Softmax activation function is a combination of multiple sigmoid activation functions. The output values from these function will be in the range 0 to 1, which can be treated as particular class data points. Unlike the sigmoid function used for binary classification, the softmax function can be used for multiple class classification [6]. Models built using the softmax activation function have an output layer with the same number of neurons as the number of classes. Softmax function can be mathematically expressed as:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} for j = 1, ..., K. \tag{2.4}$$

   Here, an exponential function is applied to each element z_i of an input vector z and normalizing these values by dividing with the sum of all the exponents. The sum of the components in the output vector $\sigma(z)$

2.6 Optimization Algorithm

Optimization algorithms are used to reduce the randomness in neural networks. Using optimization algorithms, losses are reduced by changing the attributes of the neural networks. With reduced losses, the accuracy of the result is increasing.

Adam

Adaptive Moment Estimation(also known as Adam) is a gradient-based method where the past squared gradients like Adadelta and RMSprop are accumulated and stores the decaying mean of the past gradients [9]. This method of optimization is straightforward to implement and requires less memory. This algorithm is an efficient algorithm for gradient-based optimization of stochastic objective functions.

2.7 Loss Function

The loss function depicts the failure of the model to predict valid classification. While training and testing the model, the general motto is to minimize the loss to increase the model's accuracy. The loss function used in our model is 'categorical cross-entropy loss'. For an activation function x, where x_i is the activation function for class i and y is the index of a correct label, the loss is computed mathematically as:

$$L(x,y) = -\log(\frac{e^{x_y}}{\sum_j e^{x_j}}) \tag{2.5}$$

2.8 Computer Vision

Computer vision is the transformation of data from a still or video camera into either a decision or a new representation. It is a process that helps in understanding how images and videos are stored and how to manipulate and retrieve data from them. It is mainly used for detecting and recognizing the required data from a video (or) image. OpenCV is an open-source Computer Vision library, which is helpful in detecting and extracting the image [12].

2.9 Image Processing

Image processing is a method to perform some operations on the images to enhance the image in order to scrap useful information from them. Image processing is a method in which images are sent as input, and their features or characteristics are extracted. Image processing typically has three steps:

1. Importing tools via image acquisition tools.

2. Processing the images through various processing techniques.

3. Sending an altered image or reporting that is based on image analysis as an output.

The image processing techniques used in our thesis are grayscale and histogram equalization, which are briefly explained in this thesis.

Grayscale

In digital images, grayscale images have their pixel values representing the intensity information of the light. Grayscale images only display black and white colours with varying intensities. Gray image is usually represented by 8-bit, thus ranging the pixel values from 0 to 255. To avoid the complicated process of processing colour images, all the images in the dataset are grayscaled.

Image Enhancement

Image enhancement is the primary step in image processing. The quality of the image is significantly increased by certain methods like contrasting, detailing, removing the noise in the image, sharpening the image, etc. Image enhancement is a process of manipulating the pixel intensity of the input image so that the interpretation ability or the information contained in the image can be improved [20]. The technique used in this thesis is Histogram Equalization, which is a common technique used for image enhancement. The gray levels of the image are remapped based on the probability distributions of the input image. The image's histogram is flattened and stretched, resulting in the contrast enhancement [25].

2.10 Image Augmentation

Image Augmentation is a data-space solution to the problem of overfitting in neural networks. Over-fitting is a phenomenon where a performance difference of a model can be noticed between the training and testing of the model. Over-fitting occurs when the model is trained with an excess of the same data resulting in memorizing the incorrect data. Image Augmentation decreases the validation and training error to reduce the problem of over-fitting [33]. It creates artificial images through various processing methods like zooming, shifting, random rotation, shearing etc. This is a useful technique to increase the size of the training dataset without acquiring new images.

2.11 Performance Evaluation Metrics

To identify the efficiency of a model, one needs to select proper evaluation metrics. Accuracy score can evaluate the performance of the model, which is trained for multi-level classification. In our thesis, the accuracy score is checked for the model to evaluate the model before and after parameter tuning. Given classwise accuracy A_i for class i, average accuracy is calculated as:

$$A_c = \frac{1}{n}\sum_{i=1}^{n} A_i \tag{2.6}$$

Chapter 3

Related Work

Previously various classifiers and object-classification algorithms were used for the recognition of the traffic signs. Support vector machine, random forest, neural network, deep learning models, decision fusion. [5, 23, 38] Among these algorithms, deep learning models have proven effective in many fields, including speech recognition, Natural Language Processing (NLP), and Image classification [17]. There are many deep learning classifiers like Convolution Neural Network (CNN), Deep Boltzmann Machines (DBM) for image classification [16].

Zhongqin et al. [11] had performed Traffic sign classification using improved VGG-16 architecture and attained 99% model accuracy with the German Traffic Sign Recognition Benchmark (GTSRB) [4] dataset. In their proposed model, redundant layers are removed from the VGG-16 architecture. Also, batch normalization and pooling layers were added to greatly reduce the parameters and accelerate the model's accuracy.

Shangzheng [30] proposed a fast traffic sign recognition method based on an improved CNN model by image processing and machine vision processing technology along with in-depth learning in target classification. The proposal uses HOG features to detect the traffic signs and an improved CNN model to determine the traffic sign. The proposed model accurately locate the traffic sign with improved accuracy and reduced false detection rate.

Tsoi and Wheelus [37] had performed TSR with cost-sensitive deep learning CNN model and attained 86.5% model accuracy. Their study built a model to classify traffic signals with cost-sensitive techniques like cost-sensitive rejection sampling and the use of the cost-sensitive loss functions. Their baseline model consists of 3 convolution layers, 2 max-pooling layers, 3 fully connected layers, 2 dropout layers and a softmax layer.

Shabarinath and Muralidhar [29] proposes an optimized CNN architecture based on VGGNet along with image processing techniques for traffic sign classification. They have utilized the German Traffic Sign Detection Benchmark(GTSDB) for testing and training their model. With Google's TensorFlow running in the background, they achieved 99% validation accuracy and 100% test accuracy for novel input inference. The authors applied pruning and post-training quantization for the trained model to obtain less memory footprint of the CNN.

Dhar et al. [14] had performed TSR in real-time with image acquisition techniques using colour ques from the HSV colour model. The desired area in the image is cropped using region properties and shape signature. The extracted image is thus classified using a CNN model with two convolutional layers. The authors have achieved 97% using Bangladesh Traffic signs where the extracted image is classified using CNN. The authors have also performed the classification using various other classifiers such as SVM(cubic and quadratic), ANN, Decision trees, KNN, Ensembles and achieved the highest accuracy for the CNN classifier.

Islam [21] has used LeNET architecture for the classification of the traffic signs. The author apprises two models based on CNN, one for the sign classification while the other is for shape classification. This model was not built for real-time implementation but can be helpful.

Shopa, Sumitha and Patra [32] presented a study to recognize traffic sign pattern using OpenCV techniques. In their work, they have used the HSV algorithm for image detection and Gaussian filter for image extraction. Image recognition is done by implementing knn methods. Image pre-processing, feature extraction and classification are the three modules used to recognize the traffic sign.

In the above mentioned works, either the authors use CNN for traffic sign recognition or OpenCV for traffic sign detection and extraction. Our paper proposes a CNN model to classify the traffic sign and test this model using the images extracted from real-time using OpenCV. The proposed CNN model is based on the LeNET model but has 4 convolution layers, 2 pooling layers, 2 dropout layers, followed by an output layer.

Chapter 4

Method

To achieve the aim of the thesis and to address the research questions, we employed the experimentation method in our thesis. A CNN model based on the modified LeNET model was built to conduct the experiment. GTSRB dataset is used to train and test the model to achieve good accuracy. An experiment is done by running the model after parameter tuning to find if it makes a significant change in the model's accuracy. Later, another experiment is done where the built model is used to recognize and classify the traffic sign image extracted using OpenCV methods in real-time. The procedure of this experimentation is as follows:

- Splitting up the images into train and test dataset.
- Pre-processing the images in respective datasets for enhancing the image quality and interpret ability.
- Building a cost-sensitive CNN model to classify the images.
- Tuning the parameters to find how significant the model's accuracy is improved.
- Utilizing this model to classify images extracted in real-time using OpenCV.
- Presenting the experiment results.

Figure 4.1 summarizes the steps which are conducted in the experiment to recognize traffic signs. The steps are mainly classified into two phases.

Phase 1: Training and testing the built CNN model with default parameters and then perform parameter tuning.

Phase 2: Testing the model in real-time using OpenCV.

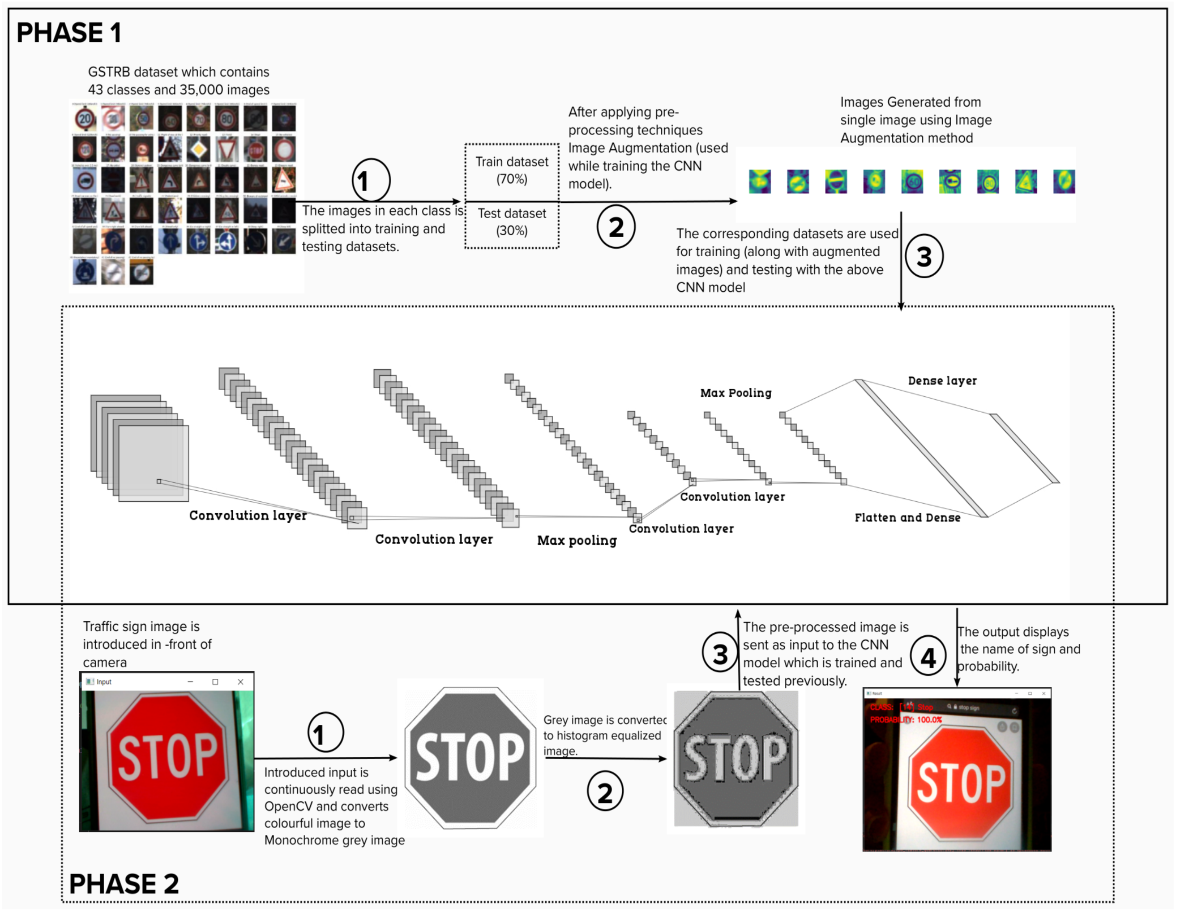
PHASE 1
GSTRB dataset which contains 43 classes and 35,000 images
1
The images in each class is splitted into training and testing datasets.
Train dataset (70%)
Test dataset (30%)
After applying pre-processing techniques Image Augmentation (used while training the CNN model).
2
Images Generated from single image using Image Augmentation method
The corresponding datasets are used for training (along with augmented images) and testing with the above CNN model
3
Convolution layer
Convolution layer
Max pooling
Convolution layer
Max Pooling
Convolution layer
Dense layer
Flatten and Dense
Traffic sign image is introduced in -front of camera
1
Introduced input is continuously read using OpenCV and converts colourful image to Monochrome grey image
Grey image is converted to histogram equalized image.
2
3
The pre-processed image is sent as input to the CNN model which is trained and tested previously.
4
The output displays the name of sign and probability.
PHASE 2

Figure 4.1: Steps involved in training, testing the proposed model (Phase 1) and testing the model in real-time (Phase 2).

4.1 Experimentation Environment

Python[1]

Python is an interpreted high-level, general-purpose, easily readable programming language. It is dynamically typed and garbage-collected. It supports procedural, functional and object-oriented programming. Python has a huge collection of libraries that provides tools suited to many tasks. Python is popular for Data Science, Machine Learning and Artificial Intelligence. It is also popular for web development as it contains good frameworks such as Flask, Django etc. The python libraries used in this thesis are described as follows.

Numpy[2]

NumPy (also called Numerical Python) is a library in python which is used for scientific computing. NumPy contains a multi-dimensional array and matrix data

[1]https://www.python.org/
[2]https://numpy.org/

structures. It can be utilised to perform a number of mathematical operations on arrays such as trigonometric, statistical, and algebraic routines.

Matplotlib[3]

Matplotlib is a cross-platform data visualization and graphical plotting library in python used for plotting attractive graphs. Numpy is the required dependency for matplotlib to plot graphs. Matplotlib script is structured so that just a few lines of code is enough to generate a visual data plot in many instances. It also provides a UI menu for customizing the plots. Matplotlib is widely known and easier to visualize among other libraries that represent data.

Keras[4]

Keras is a Google-developed high-level deep learning API for building neural networks. It is used to make the implementation of neural networks easy. It also supports multiple back-end neural network computation. Keras is easy to learn and use, as it provides python front-end with high-level of abstraction, having multiple back-end options for computation purposes.

OpenCV[5]

OpenCV is an open-source library for Image processing and Computer Vision. It is used to detect and recognize entities in real-time. When it is integrated with other libraries such as Numpy, it is capable of preprocessing the OpenCV array structure for analysis, i.e. the detected image with the help of OpenCV can be used for preprocessing using Numpy. It supports C++, Java and Python interfaces.

Pandas[6]

Pandas is an Open-source package used for Data analysis and manipulation. It is released under the BSD license. It is built on top of the Numpy package. Panda saves a lot of time by removing time-consuming and repetitive tasks associated with data.

Scikit-learn[7]

Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms. It is built on packages like Numpy, Pandas and Matplotlib.

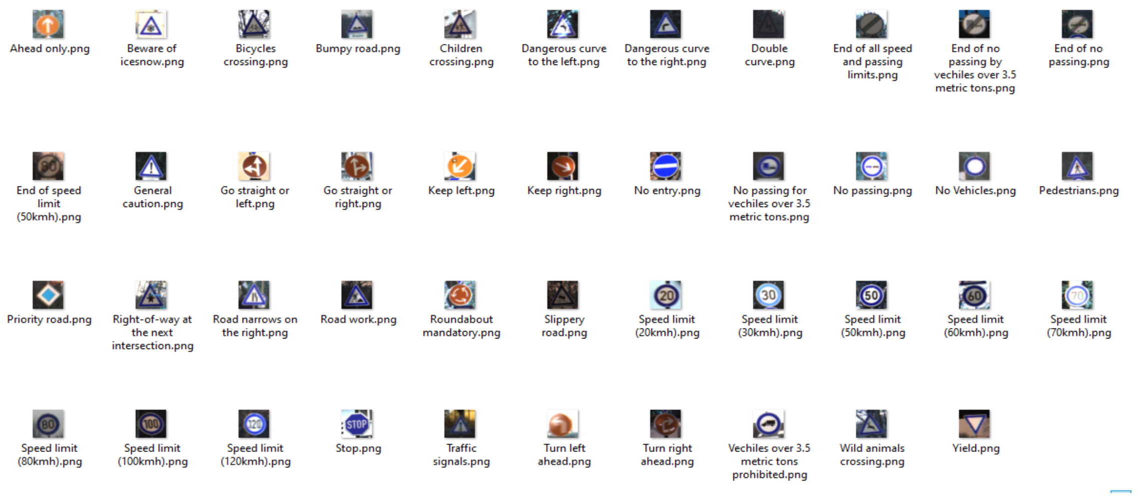

Figure 4.2: Sample of the GTSRB dataset

4.2 Data Overview

German Traffic Sign Recognition Benchmark (GTSRB) is the dataset considered for this experimentation process. This dataset was introduced in International Joint Conference On Neural Networks (IJCNN) 2011 [36]. It is a benchmark dataset for TSR, which is multi-class and used for single-image classification. The dataset was used in the IJCNN competition, and high classification accuracy was achieved by using CNN models as well as manual classification. The dataset contains over 35,000 images of different traffic signs. It is further classified into 43 different classes. These 43 class labels are the traffic signs which are common in most the countries. A sample image of each class can be shown in the Figure 4.2. The dataset is quite varying. Some of the classes have many images, while some classes have few images. The size of the dataset is around 300 MB [1]. The dataset is divided into train and test datasets. 70% of images of every class are considered for training and remaining used for testing and validation. Step 1 of phase 1 in the Figure 4.1 is done here. The Table 4.1 refers to the number of images are there in each dataset (Training, Testing and Validation). There is a CSV file containing the labels for the corresponding class and class Id along with the dataset.

Category	Total Images	Images for Training	Images for Testing	Images for Validation
GSTRB dataset	34,799	24,360	10,440	10,440

Table 4.1: Distribution of images across different datasets

[3]https://matplotlib.org/
[4]https://keras.io/
[5]https://opencv.org/
[6]https://pandas.pydata.org/
[7]https://scikit-learn.org/stable/

4.3 Processing The Dataset Images

Step 2 of phase 1 in the Figure 4.1 is done here. After splitting the dataset for training and testing, the preprocessing techniques are applied on training and testing datasets, respectively, by iterating through the images. Initially, the image is converted into an array of pixels. The array of pixels are sent as input to gray-scale method. Gray-scale method is used to convert colourful image to monochrome Grayscale image. The conversion from a RGB image to gray is done with **cvtColor(src, CV_RGB2GRAY).**

- **cvtColor** - It is an OpenCV method which converts an image from one color space to another.
- **src** - Input image
- **CV_RGB2GRAY** - Transforms RGB colors to gray using the formula.

$$\mathbf{Y = 0.299\ R + 0.587\ G + 0.114\ B} \tag{4.1}$$

Where **R, G, B** are Red, Green and Blue.

By following the above-mentioned proportions of Red, Green and Blue colours, the Grayscale method transforms into a monochrome gray image.

Now the converted grayscaled array is sent as input to the Histogram equalization method. This method is used to improve contrast in images by spreading out the most frequent intensity values. Initially, the pixel values are higher in some areas and lower in some areas. What Histogram Equalization does is to stretch out the pixel range by uniformly distributing the pixel values respectively. It uses the Cumulative Distribution Function (CDF) for re-mapping the Gray-scale distribution to Histogram Equalization distribution.

$$H1(i) = \sum_{0<j<i} H(j) \tag{4.2}$$

After the Histogram equalization method, the array of pixels are divided by 255 so that the pixel values are between 0 to 1, respectively. The Figure 4.3 shows sample pre-processed images.

4.4 Image Augmentation

Image augmentation is a useful method for building the CNN model by increasing the dataset images (step 3 of phase 1 in the Figure 4.1). It can create more images from a single image so that the CNN model can have a lot of images for training the model. The idea is simple, it creates multiple duplicate images from a single image with some variations to that. Keras provides a method called **ImageDataGenerator** for performing image argumentation. It has some parameters (like zoom, rotation etc.) and based on these parameters the multiple images are created. In our model, we used the following parameters for generating multiple images.

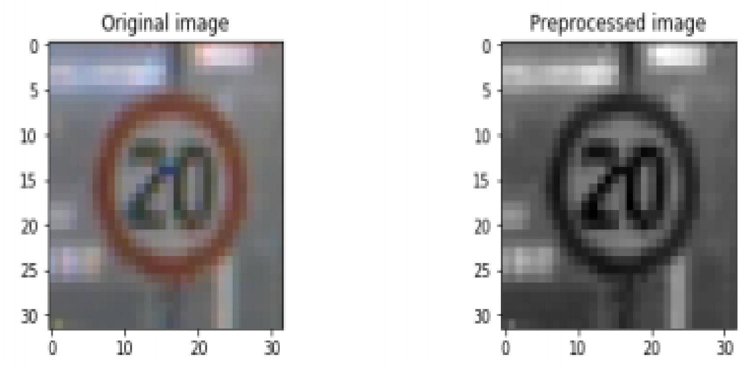

Figure 4.3: Sample Pre-processed Image

- **width_shift_range:** It shifts the image to the left or right(horizontal shifts). If the value is floating and <=1, it will take the percentage of total width as a range. We have taken the value as 0.1, i.e. it will take 10% of the total width as a range.

- **height_shift_range:** It works same as **width_shift_range** but shift vertically(up or down)

- **zoom_range:** It zooms the image to the mentioned percentage. We have given the value 0.2

- **Shear_range:** Shear means that the image will be distorted along an axis, this parameter is used to augment images in a way how humans see things from different angles. The value here is the angle distorted along axes. We have given the value as 0.1.

- **Rotation_range:** It rotates the image by the given angle. We have given the angle as 10 degrees.

With the help of the above parameters, 20 images are generated from an image (The limit is set to 20). These generated images are used for training the CNN model as well. The Figure 4.4 shows sample augmented images.

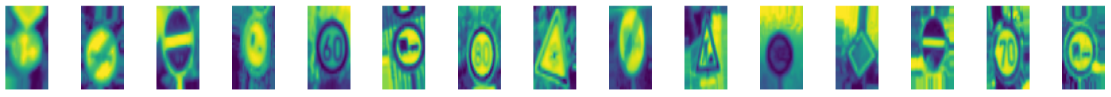

Figure 4.4: Augmented Images

Type of layer	Output shape	Parameters
conv2d (Conv2D)	(None, 28, 28, 60)	1560
conv2d_1 (Conv2D)	(None, 24, 24, 60)	90060
max_pooling2d (MaxPooling2D)	(None, 12, 12, 60)	0
conv2d_2 (Conv2D)	(None, 10, 10, 30)	16230
conv2d_3 (Conv2D)	(None, 8, 8, 30)	8130
max_pooling2d_1 (MaxPooling2)	(None, 4, 4, 30)	0
dropout (Dropout)	(None, 4, 4, 30)	0
flatten (Flatten)	(None, 480)	0
dense (Dense)	(None, 500)	240500
$\text{dropout}_1(Dropout)$	(None, 500)	0
$\text{dense}_1(Dense)$	(None, 43)	21543

Table 4.2: Architecture of CNN

4.5 Building The CNN Model

In our experiment, we have taken lenet model based on [26] and made some changes to it. LeNet-5 CNN architecture is made up of 7 layers. The layer composition consists of 3 convolutional layers, 2 subsampling layers and 2 fully connected layers. We added 1 more convolutional layer and 2 dropout layers to lenet model. So, in our experiment, we used 4 convolution layers, 2 max-pooling layers, 2 dropout layers, 2 Fully-Connected layers, and 1 flatten method (which is shown in the Figure 4.5). We used 60 kernels of size 5*5 in the first and second Convolution layers with ReLU activation. For the 3rd and 4th layers, 30 kernels of size 3*3 are used. For both max-pooling layers, we used pool_size as 2*2 filter. 2 drop out layers are used with 0.5 drop, i.e. 50% neurons will shut down at each training step by zeroing out the neuron values. 2 dense layers for which one dense layer performs 'relu' activation and other performs 'softmax'.

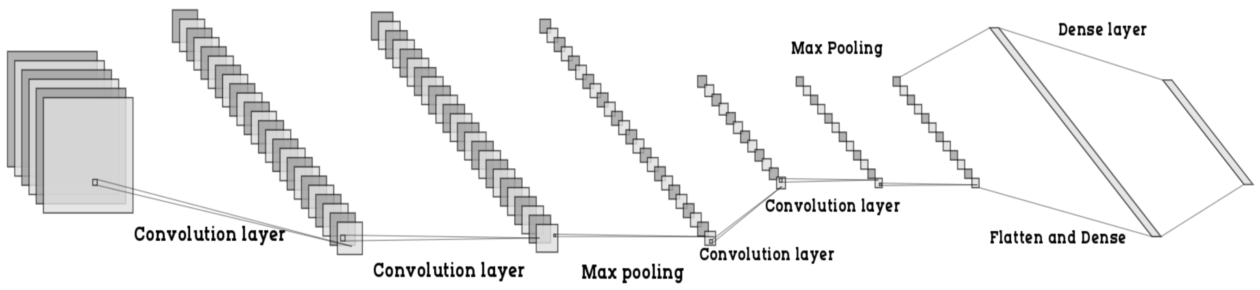

Figure 4.5: Our proposed CNN model Architecture

1. In the conv2d layer, element-wise multiplication is performed with the pre-processed 3D image array and kernel (or) filter of size 5*5, sums up the array

and updates the sum value in the corresponding location. In the same way, every pixel value of the image array gets updated. 60 filters are applied in this layer. After this layer, the size changes to 28*28. Later, the RELU layer applies an activation function max (0, x) on all the pixel values of an output image.

2. Same operation is performed on conv2d_1 layer with "relu" activation function. The size will be reduced to 24*24.
3. Later, max pooling is applied, which takes the maximum element from the region of the feature map covered by the filter. The main purpose of the max pool is to get the most prominent features of the previous feature map. Here, a filter size of 2*2 is given.
4. After max pooling, 2 conv_2d layers applied one after another with 30 filters of size 3*3 and with "relu" activation function.
5. Again max-pooling applied with 2*2 filter size. Later, a drop out of 0.5 is used to reduce the connections to half. After performing drop out, flatten method is used, which converts the image array into a single dimension.
6. Finally, 2 fully connected layers are used (one with "relu" activation function and the other with "softmax" activation function) with one dropout in between (0.5) to get the final output array.

After building, the model has compiled with adam optimizer along with accuracy metric and loss.

4.6 Training and Testing The Model

Training of the model is done using Adam optimizer and categorical_cross entropy as loss function. Step 3 of phase 1 in the Figure 4.1 is done here. As shown in the Figure 4.6 the number of images are few in some classes, so data augmentation is used while training the model to generate multiple duplicate images from a single image. Fit method is used to train the model, which contains the following parameters.

- Augmented images generated from flow method of image data generator class per each image in training dataset.
- **steps_per_epoch:** len(X_train)/ batch_size
- **epochs:** Initially 10 epochs are taken, later can be changed for parameter tuning.
- **Validation_data:** Validation is also performed along with train, using validation dataset.

After performing the training, graphs are plotted for loss and accuracy with increasing epochs among training and validation datasets. After training is completed, testing is performed with the testing dataset. These are the images that are not used for training, i.e., unseen traffic images. Later, the model is saved using the save method in '.h5' extension.

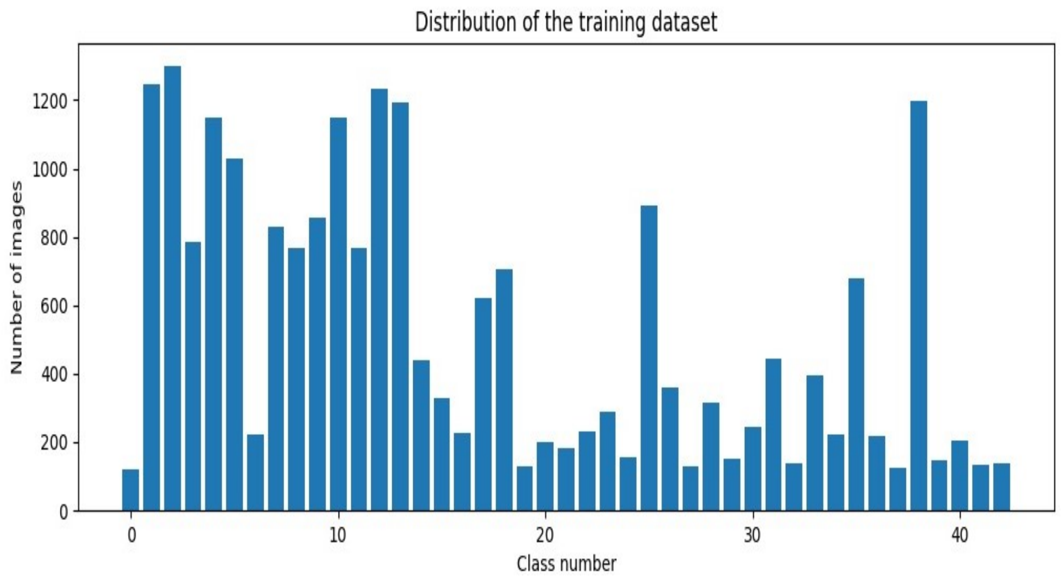

Figure 4.6: Distribution of the training dataset

4.7 Parameter Tuning

After testing the model, we performed parameter tuning, in which we performed a systematic test with multiple combinations of epochs and learning rate. There are a lot of hyper-parameters in the deep learning model. Among these parameters, epochs and learning rate significantly affect the models accuracy. The learning rate controls how much to change the model when the model's weights are updated. Learning rate is considered as one of the important hyper-parameter that affects the model's accuracy [34]. Epoch is defined as a full pass of the dataset. Tuning the epochs may significantly affect the accuracy. Hence both these parameters are used for tuning. The combinations are summarized in Table 4.3.

Learning rate	**epochs**
0.001	10
0.001	20
0.002	10
0.002	20
0.001	30
0.0005	20
0.001	5

Table 4.3: Various combinations of parameter tuning

4.8 Testing The Model In Real-time

Later, OpenCV is used to set up the live camera to introduce the image to the camera as input. The image is extracted using OpenCV and converted to an array of pixels

(Figure 4.1). Later the array is resized into a 32*32 dimension array using resize method (OpenCV method). The converted array will be pre-processed using the grayscale method (step 1 of phase 2 of Figure 4.1), histogram equalization method (step 2 of phase 2 of Figure 4.1), and sent as input to the saved CNN model, which was trained and tested previously (step 3 of phase 2 of Figure 4.1 is mentioned here). Here, we test our model on a real-time video captured using the webcam or any other external feed. After we access the web camera, every frame in the live video is processed. We use our saved model to predict the probability of the processed image. If the probability of the prediction is higher than the threshold value, i.e., 0.75, then the image is recognised as a traffic sign. A corresponding class name and the prediction probability value are displayed (step 4 of phase 2 of Figure 4.1 is mentioned here). The probability value depends on the size of the training dataset and on the architecture of the model. If the dataset of the corresponding class is large in size, then we get a high probability prediction value to that class images.

Chapter 5

Results and Analysis

This chapter presents the results obtained during the experimentation method. The results include the evaluation of the model before and after parameter tuning. The evaluation metrics used in the thesis is accuracy. Also, a sample classification of the image introduced to a real-time camera is presented with its class name and prediction probability.

5.1 Experiment-1: Training The Model

In this experiment, a CNN model with a modified LeNet model is built with 4 convolution layers, 2 max-pooling layers, 2 dropout layers, 2 dense layers, and 1 flatten layer.

5.1.1 The model trained with default parameter values

The model runs on the default parameter values configured, i.e., 0.001 for the learning rate and 10 epochs. The figure 5.1 represents the accuracy of the model for train and test data and 5.2 represents the loss of the model for train and test data.

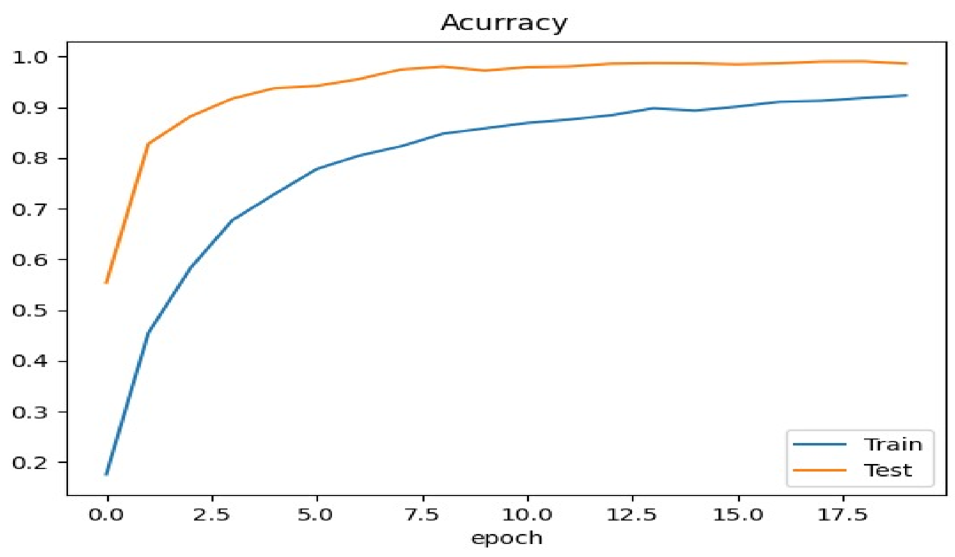

Figure 5.1: Model accuracy with default parameter values

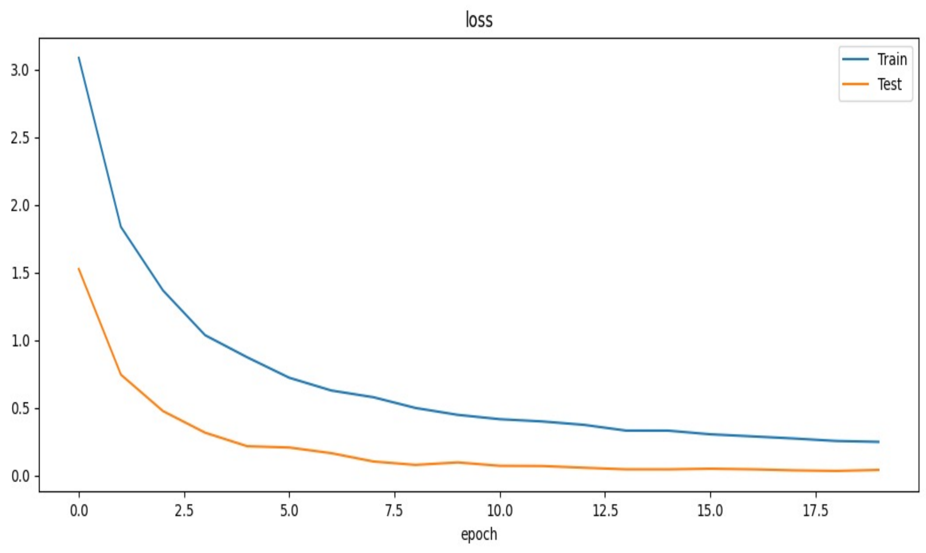

Figure 5.2: Loss occurred

5.1.2 The model trained after parameter tuning with various combinations

In this experiment, we tune the parameters learning rate and epochs, which leads to a significant improvement in the model's accuracy. We have implemented various combinations of these parameters to achieve significant model accuracy. The below graphs depict the accuracy and loss of the model after the tuning of the parameters. The 5.11 to 5.3, blue line indicates the train data, and the orange line indicates the test data.

For learning rate = 0.002 and epochs = 10

Figure 5.3 and 5.4 represent the accuracy and loss of the model for train and test data when the learning rate is tuned to 0.002 and epochs to 10.

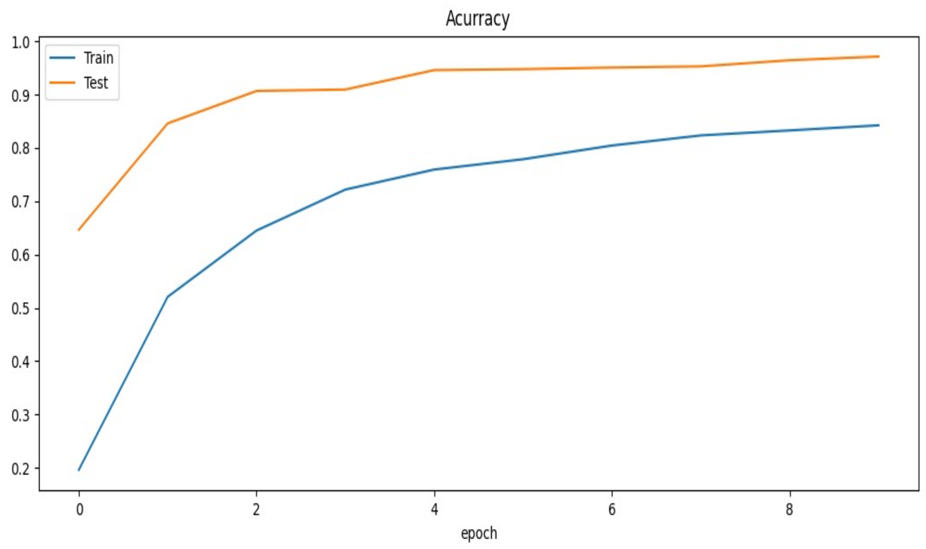

Figure 5.3: Accuracy

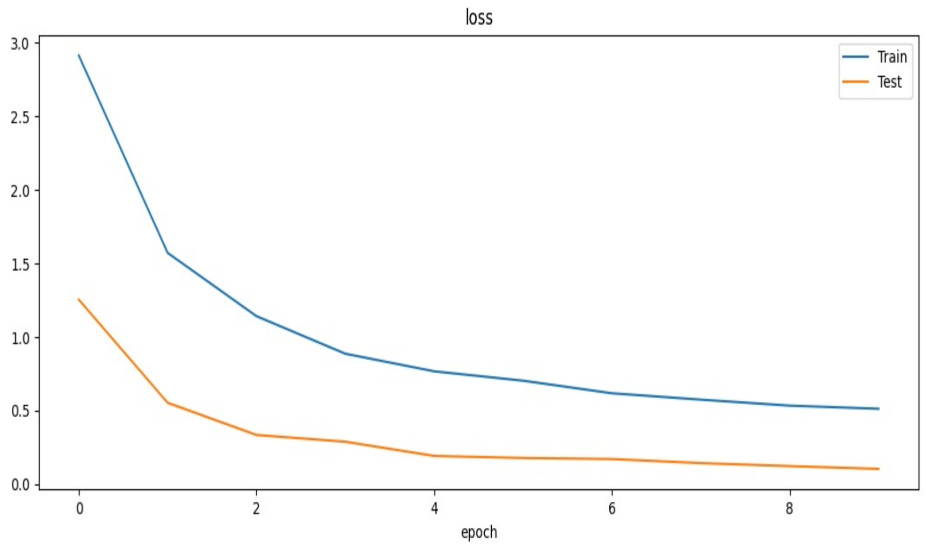

Figure 5.4: Loss

For learning rate = 0.001 and epochs = 20

Figures 5.5 and 5.6 represent the accuracy and loss of the model for train and test data when the learning rate is tuned to 0.001 and epochs to 20.

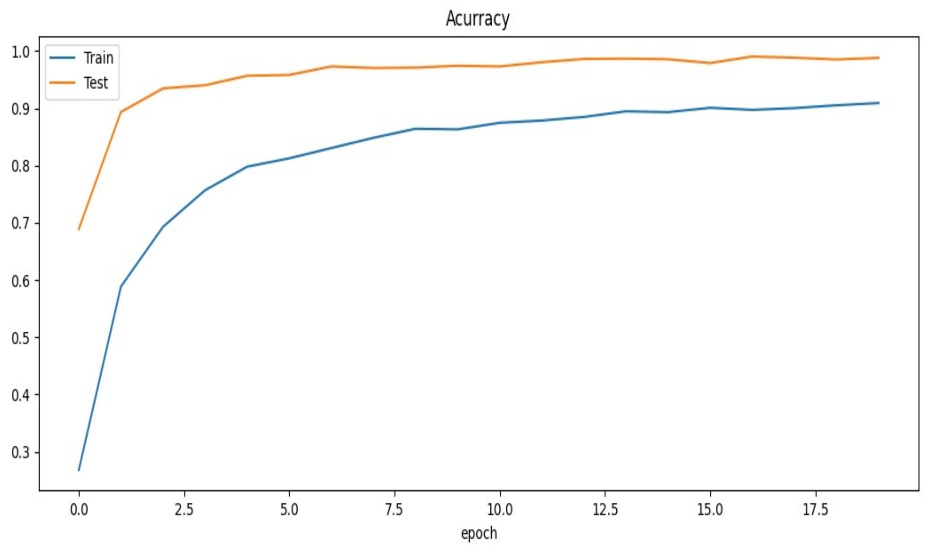

Figure 5.5: Accuracy

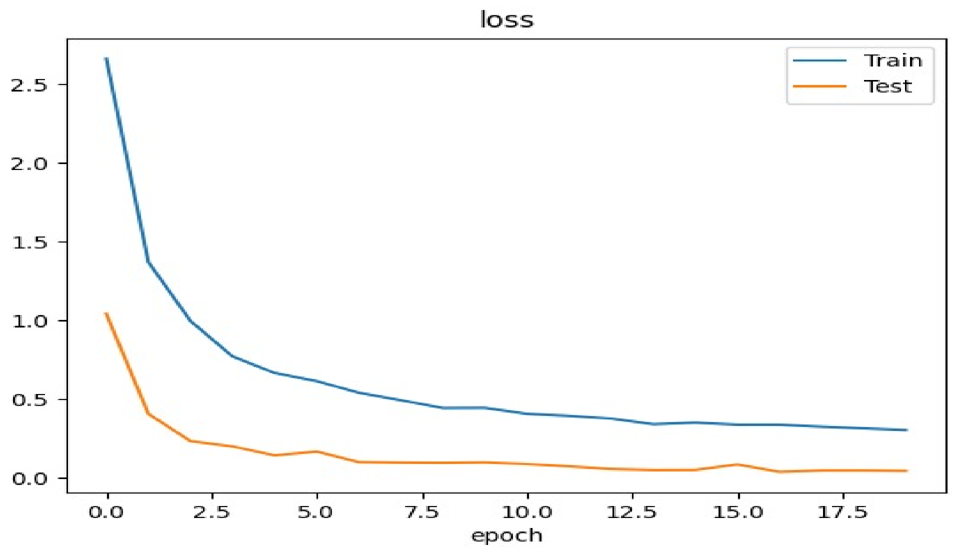

Figure 5.6: Loss

For learning rate = 0.002 and epochs = 20

Figures 5.7 and 5.8 represent the accuracy and loss of the model for train and test data when the learning rate is tuned to 0.002 and epochs to 20.

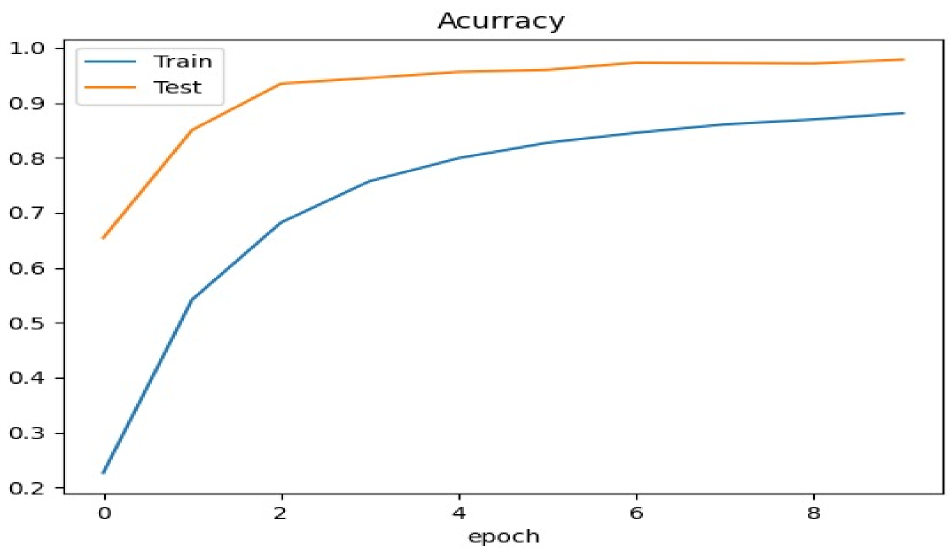

Figure 5.7: Accuracy

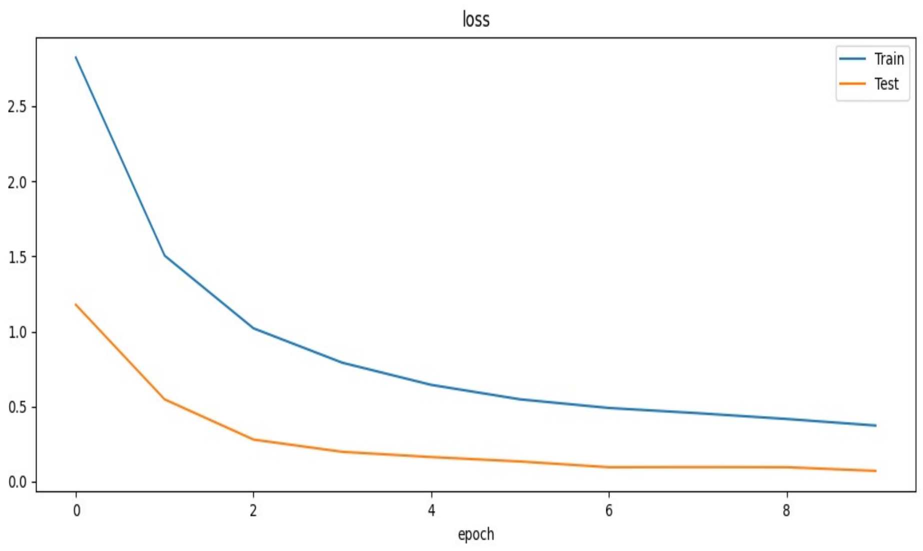

Figure 5.8: Loss occurred

For learning rate = 0.0005 and epochs = 20

Figures 5.9 and 5.10 represent the accuracy and loss of the model for train and test data when the learning rate is tuned to 0.0005 and epochs to 20.

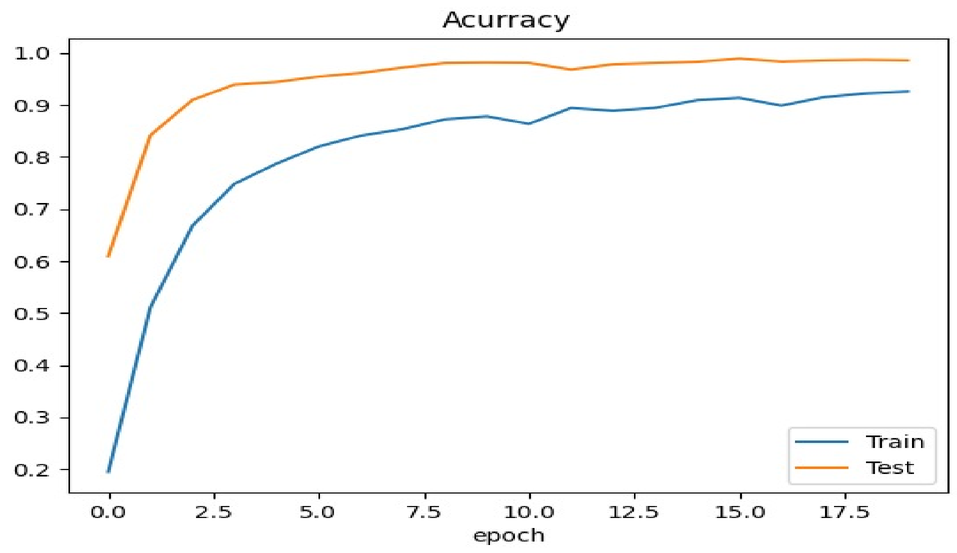

Figure 5.9: Accuracy

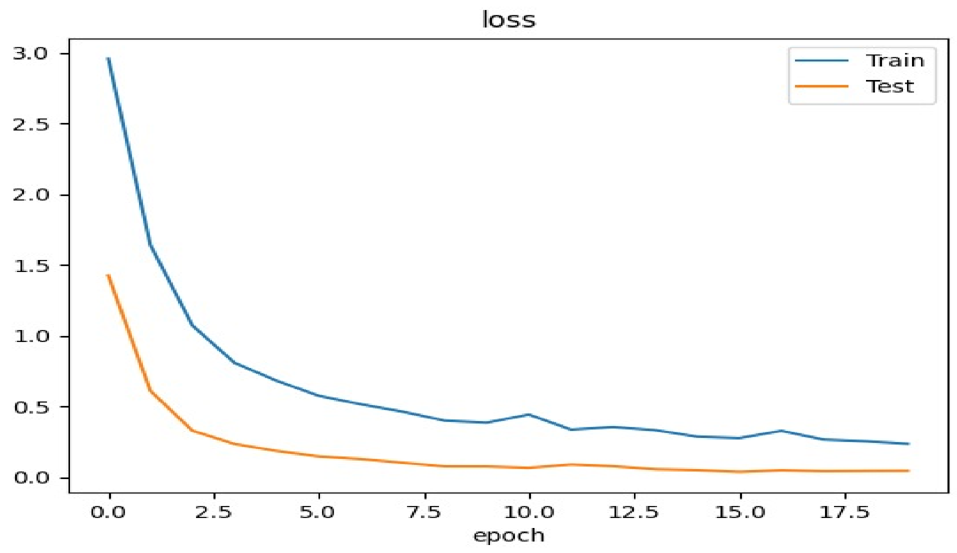

Figure 5.10: Loss occurred

For learning rate = 0.001 and epochs = 5

Figures 5.11 and 5.12 represent the accuracy and loss of the model for train and test data when the learning rate is tuned to 0.001 and epochs to 5.

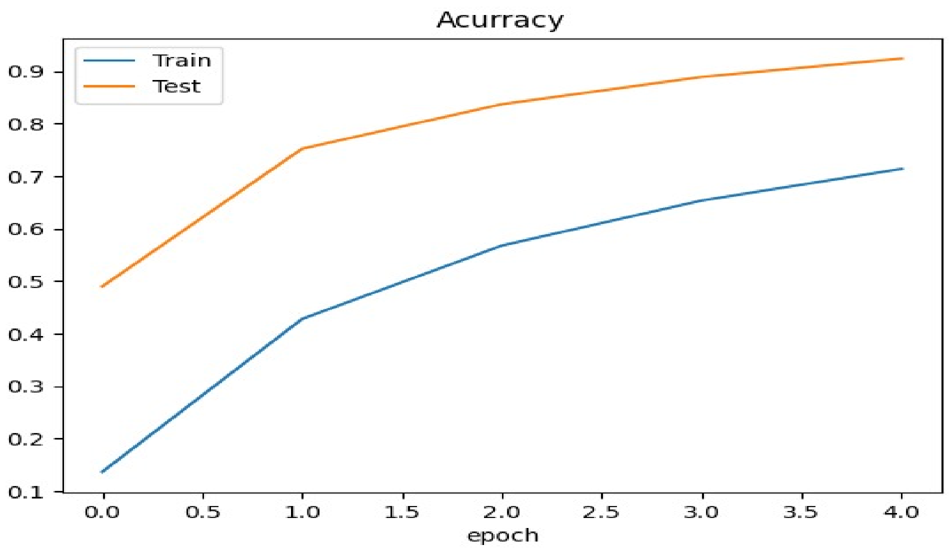

Figure 5.11: Accuracy

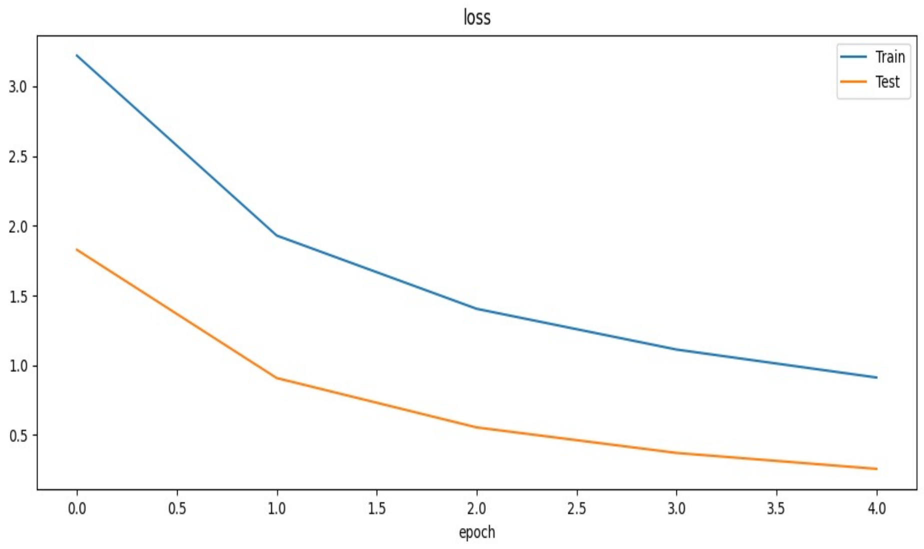

Figure 5.12: Loss occurred

For learning rate = 0.001 and epochs = 30

Figures 5.13 and 5.14 represent the accuracy and loss of the model for train and test data when the learning rate is tuned to 0.001 and epochs to 30.

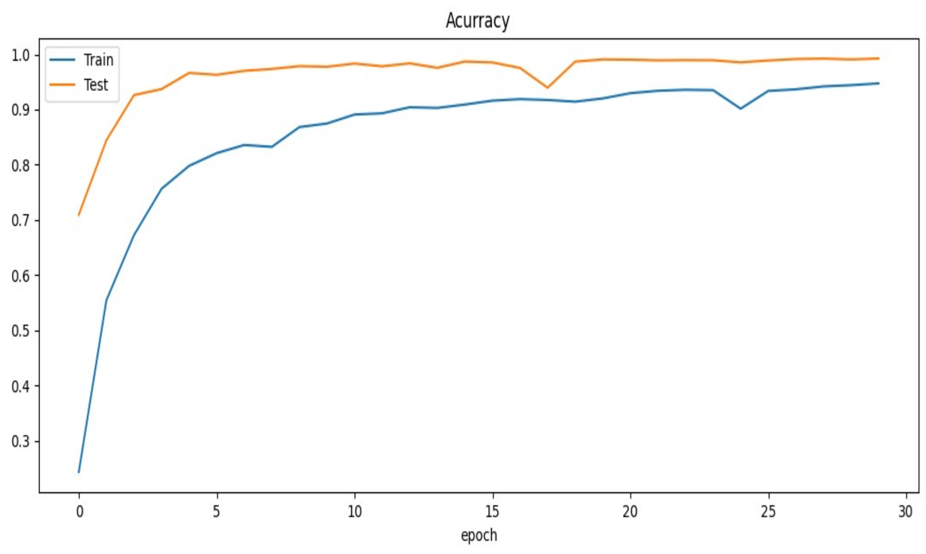

Figure 5.13: Accuracy

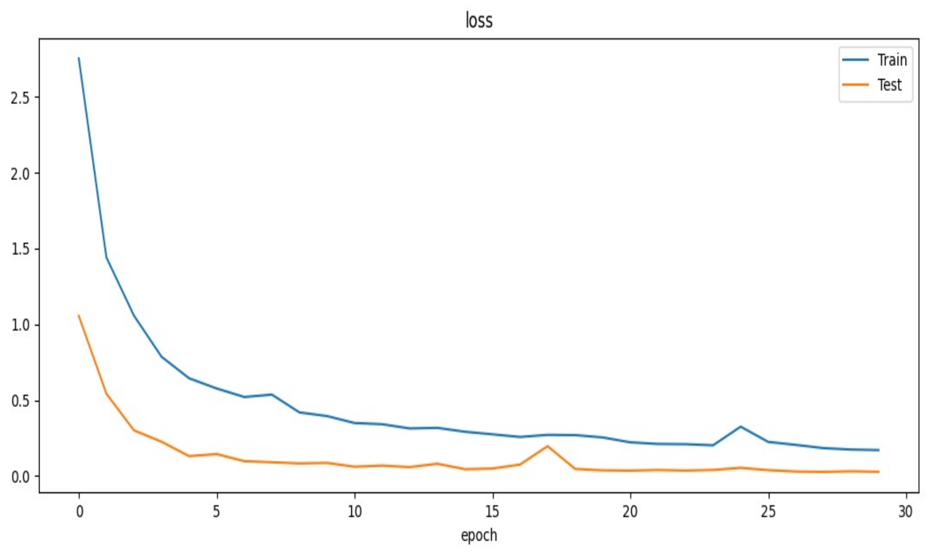

Figure 5.14: Loss occurred

5.2 Experiment-3: Using The Model In Real-time

In this experiment, we tested the model on a real-time video using a webcam or any external live feed. Traffic sign images introduced to the camera are new images and are not from training or testing data. New images from each class are used for testing. We process every frame of the video to check for a traffic sign. If the probability value of the prediction is higher than the threshold value, then the class label will be displayed along with the probability. The figreffig:webcam displays an example where a traffic stop sign image is placed in front of the webcam. The class name is displayed along with the probability of prediction. Each traffic sign in 43 classes is tested in real-time. While some show probability scores greater than 90%, some fluctuate their probability scores. For example, when the traffic sign 'Speed limit (20km/h)' image is placed in front of the camera, the probability score fluctuates. We tested the model with 43 different Traffic signs, one image for each class, and we

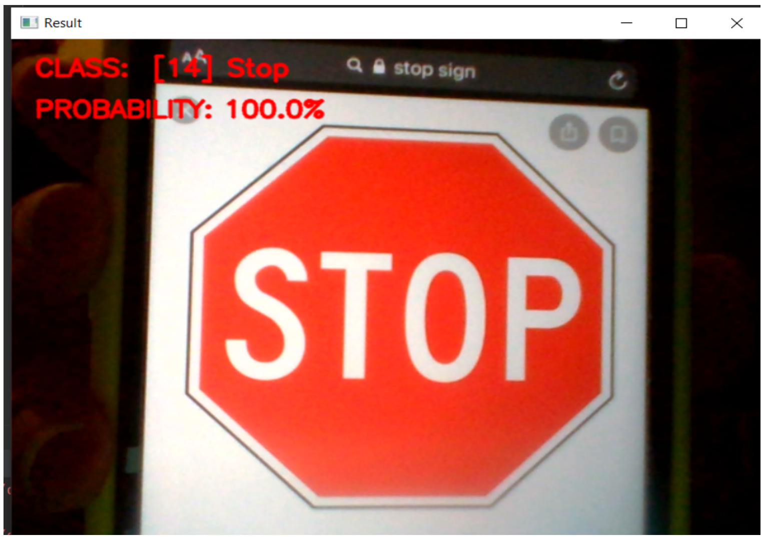

Figure 5.15: Traffic sign classification in real time using webcam

noted the probabilities. The probabilities for each class label is shown in Table 5.1.

Class Id	Traffic sign	Probability
0	Speed limit (20km/h)	80-90%
1	Speed limit (30km/h)	100%
2	Speed limit (50km/h)	98%
3	Speed limit (60km/h)	99%
4	Speed limit (70km/h)	100%
5	Speed limit (80km/h)	100%
6	End of speed limit (80km/h)	95-98%
7	Speed limit (100km/h)	99%
8	Speed limit (120km/h)	98%
9	No passing	100%
10	No passing for vechiles over 3.5 metric tons	99%
11	Right-of-way at the next intersection	100%
12	Priority road	97-99%
13	Yield	99%
14	Stop	92-95%
15	No vehicles	91-95%
16	Vehicles over 3.5 metric tons prohibited	93-96%
17	No entry	98-99%
18	General caution	99%
19	Dangerous curve to the left	80-90%
20	Dangerous curve to the right	85-90%
21	Double curve	89-93%
22	Bumpy road	86-91%
23	Slippery road	87-90%
24	Road narrows on the right	90-93%
25	Road work	98-99%
26	Traffic signals	94-96%
27	Pedestrians	86-90%
28	Children crossing	89-94%
29	Bicycles crossing	85-90%
30	Beware of ice/snow	91-95%
31	Wild animals crossing	96-98%
32	End of all speed and passing limits	89-93%
33	Turn right ahead	95-97%
34	Turn left ahead	88-91%
35	Ahead only	99-100%
36	Go straight or right	89-92%
37	Go straight or left	85-88%
38	Keep right	100%
39	Keep left	85-88%
40	Roundabout mandatory	86-89%
41	End of no passing	90-92%
42	End of no passing by vechiles over 3.5 metric tons	87-91%

Table 5.1: Probability scores achieved when testing the model with new images in real time.

Chapter 6

Discussion

We have proposed a CNN model whose architecture is based on the LeNet model that can classify the traffic signs using the GTSRB dataset. The research questions are answered by drawing certain conclusions from the results obtained.
RQ1:
Considering the initial parameter setups of the proposed model, to what extent can parameter tuning improve the model's accuracy?
Answer:

The Figures 5.1 to 5.14 from the Chapter refchp:results reflects the accuracy and loss of train and test data. This data is obtained by tuning epochs and learning rate with various combinations. Every combination of the parameter tuning showed a considerable change in the accuracy of the model. The tablereftab:Tuning represents the values obtained from the experiment-1 in Chapter5.

Learning rate	Epochs	Train accuracy	Train Loss	Test Accuracy	Test loss
0.001	10	0.8883	0.3659	0.9843	0.0669
0.002	10	0.8549	0.4684	0.9697	0.0996
0.001	20	0.9262	0.2352	0.9857	0.0477
0.002	20	0.9052	0.3155	0.9883	0.0469
0.0005	20	0.9222	0.2508	0.9870	0.0407
0.001	5	0.7026	0.9544	0.9274	0.2580
0.001	30	0.9510	0.1664	0.9931	0.0252

Table 6.1: Accuracy and loss of model for train and test data by tuning various combinations of learning rate and epochs

Some interesting trends in the model's accuracy is found when the learning rate is tuned. Learning rate or step size is referred to as the number of weights that are updated while training the model. Generally, the learning rate is a configurable hyperparameter that is used while training the model whose range is between 0.0 - 1.0. From the table6.1, we can find that changing the learning rate from the default set optimum value, i.e. 0.001 decreases the model's accuracy. This can be caused because the learning rate controls the rate of adaption of a model to a problem. So, a large learning rate makes the model learn faster but results in a sub-optimal set of weights. Loss on the training dataset occurs due to the divergent weight decreasing the accuracy. A lower learning rate allows the model to learn an optimal set of weights but may take a lot of time. The model may be held on a sub-optimal solution. Hence, changing the learning rate from the default value decreases the model accuracy.

On the other hand, the Table 6.1 clearly depicts that increase in the number of epochs results in achieving high accuracy. But increasing the number of epochs more than necessary may lead to an overfitting problem. When the number of epochs for training the model is increased rapidly, the model learns the patterns of the sample data to a large extent. This makes the model less efficient on test data. Due to overfitting, the model shows high accuracy on the train data but fails to do so on test data. A fewer number of epochs result in underfitting. This results in low accuracy as well. The figurereffig:epoch represents the model learning patterns with the different number of epochs.

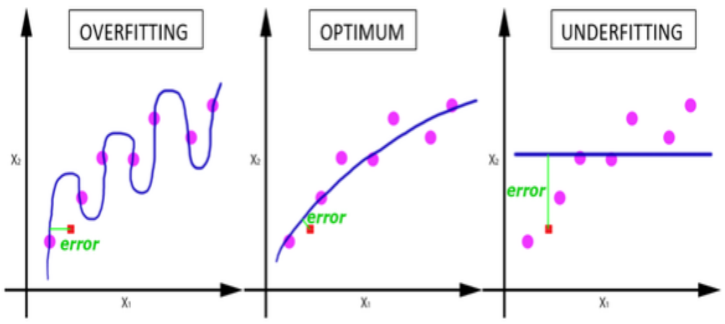

Figure 6.1: Graph plot reflecting the model learning patterns with decreasing number of epochs. (The figure is borrowed from [3])

From the above discussion, increasing the epochs significantly increase the model's accuracy up to a certain extent. With optimum learning rate set to 0.001 and epochs set to 30, 95% accuracy is achieved. A minimum loss of 0.1664 is achieved.

RQ2:

How can a traffic sign image be detected and extracted effectively when introduced to the live camera using OpenCV?

Answer:

From Experiment-2 in Chapter5, traffic sign images are introduced to the model via a web camera. The images used for testing the model are new images, and they are not from training and testing datasets. Every frame of the video is checked for a traffic sign. A pre-processed image (grayscaled and histogram equalized) is captured from the camera and uses the CNN model to classify the digital image. The probability score of the prediction is calculated and is checked with the threshold value, i.e. 0.75. This means if the probability of predicting the correct traffic sign is more than 0.75, then the corresponding class label is displayed. For a typical binary classification default threshold is 0.5, and for a multi-class classification problem, the default value ranges between 0 to 1. Employing default threshold value may not represent the optimal interpretation of the predicted probabilities. Hence a reasonably high threshold than the default threshold is employed.

While conducting Experiment 2, some traffic signs, when introduced to the camera, show fluctuating probability score as shown in Table 5.1. This is due to the unbalanced data in various classes. If the class contains optimum number of images, i.e. up to 500 images, then a high probability score is achieved. Images can be augmented into those classes whose images are insufficient to achieve high probability or a dataset containing balanced data in its classes can also be used to train the model to achieve high probability scores of the traffic sign images. This answers the second research question.

Chapter 7

Conclusions and Future Work

7.1 Conclusion

Traffic Sign Recognition helps novice drivers through Driver Assistance System and provide safe, efficient navigation in the case of self-driving vehicles. In this thesis, we proposed a cost-sensitive CNN model in terms of computational cost, which can classify the traffic signs images from the GTSRB dataset with 95% accuracy. The proposed model architecture has 4 convolution layers which have low computational cost than various state-of-the-art architectures like VGG-16, where there are 16 convolution layers. The learning rate and number of epochs were tuned in different combinations to achieve the highest accuracy possible. Later, the model was used to predict the class labels for the traffic sign images that are introduced to the web camera with reasonably high probability.

From the results, we can conclude that tuning the epoch significantly impacts the model's accuracy. The results show that changing the learning rate decreases the model's accuracy. Hence with learning rate = 0.001 and epoch = 30, the model has achieved high accuracy, i.e., 95.1%

7.2 Future work

As for future work, more parameters can be tuned, like batch-size, dropout rate etc., to show significant improvement in the model's accuracy. It would be of great interest to find out the parameters that can increase the model's accuracy when they are tuned. To solve the problem of long training times, Amazon Web Server (AWS) Deep Learning Amazon Machine Images (AMI) can be used to generate auto-scaled clusters of GPU for large scale training. One can quickly launch an Amazon EC2 instance which is pre-installed with popular deep learning frameworks and interfaces to train the model [1].

To improve the prediction probability of the images in real-time, the larger dataset can be used to train the model with more images in each class. Employing a larger dataset may result in better predictions because more features of each class label can be obtained. Oversampling the minority classes can also help in increasing the probability. The simplest way to perform oversampling is to duplicate the existing images in the minority classes so that the model will learn no extra information. Data augmentation can also be done more on the minority classes to increase the dataset size.

The model can be improved further to use in real-time. This model can be employed in the ADS system with the output as a voice prompter.

Bibliography

[1] "Amazon Deep Learning AMIs." [Online]. Available: https://aws.amazon.com/machine-learning/amis/

[2] "deep learning - Why convolutions always use odd-numbers as filter_size." [Online]. Available: https://datascience.stackexchange.com/questions/23183/why-convolutions-always-use-odd-numbers-as-filter-size

[3] "Does increasing epochs increase accuracy? - Quora." [Online]. Available: https://www.quora.com/Does-increasing-epochs-increase-accuracy

[4] "Public Archive: daaeac0d7ce1152aea9b61d9f1e19370." [Online]. Available: https://sid.erda.dk/public/archives/daaeac0d7ce1152aea9b61d9f1e19370/published-archive.html

[5] A. Adam and C. Ioannidis, "Automatic road-sign detection and classification based on support vector machines and hog descriptors." *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 2, no. 5, 2014.

[6] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi, "Learning activation functions to improve deep neural networks," *arXiv preprint arXiv:1412.6830*, 2014.

[7] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*. Ieee, 2017, pp. 1–6.

[8] J. C. Ang, A. Mirzal, H. Haron, and H. N. A. Hamed, "Supervised, unsupervised, and semi-supervised feature selection: a review on gene selection," *IEEE/ACM transactions on computational biology and bioinformatics*, vol. 13, no. 5, pp. 971–989, 2015.

[9] P. Bahar, T. Alkhouli, P. Jan-Thorsten, C. J.-S. Brix, and H. Ney, "Empirical investigation of optimization algorithms in neural machine translation," *The Prague Bulletin of Mathematical Linguistics*, vol. 108, no. 1, p. 13, 2017.

[10] Y. Bengio, I. Goodfellow, and A. Courville, *Deep learning.* MIT press Massachusetts, USA:, 2017, vol. 1.

[11] Z. Bi, L. Yu, H. Gao, P. Zhou, and H. Yao, "Improved VGG model-based efficient traffic sign recognition for safe driving in 5G scenarios," *International Journal of Machine Learning and Cybernetics*, Aug. 2020. [Online]. Available: https://doi.org/10.1007/s13042-020-01185-5

[12] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.

[13] P. Cunningham, M. Cord, and S. J. Delany, "Supervised learning," in *Machine learning techniques for multimedia.* Springer, 2008, pp. 21–49.

[14] P. Dhar, M. Z. Abedin, T. Biswas, and A. Datta, "Traffic sign detection — a new approach and recognition using convolution neural network," in *2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, 2017, pp. 416–419.

[15] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning.* MIT press Cambridge, 2016, vol. 1, no. 2.

[16] H. Guan, W. Yan, Y. Yu, L. Zhong, and D. Li, "Robust Traffic-Sign Detection and Classification Using Mobile LiDAR Data With Digital Images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 11, no. 5, pp. 1715–1724, May 2018, conference Name: IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing.

[17] J. Han, D. Zhang, G. Cheng, L. Guo, and J. Ren, "Object detection in optical remote sensing images based on weakly supervised learning and high-level feature learning," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 6, pp. 3325–3337, 2014.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[19] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: The german traffic sign detection benchmark," in *The 2013 international joint conference on neural networks (IJCNN).* Ieee, 2013, pp. 1–8.

[20] H. Ibrahim and N. S. P. Kong, "Brightness preserving dynamic histogram equalization for image contrast enhancement," *IEEE Transactions on Consumer Electronics*, vol. 53, no. 4, pp. 1752–1758, 2007.

[21] M. T. Islam, "Traffic sign detection and recognition based on convolutional neural networks," in *2019 International Conference on Advances in Computing, Communication and Control (ICAC3)*, 2019, pp. 1–6.

[22] M. B. Jensen, M. P. Philipsen, A. Møgelmose, T. B. Moeslund, and M. M. Trivedi, "Vision for looking at traffic lights: Issues, survey, and perspectives," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 7, pp. 1800–1815, 2016.

[23] J. Jin, K. Fu, and C. Zhang, "Traffic sign recognition with hinge loss trained convolutional neural networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 1991–2000, 2014.

[24] J. D. Kelleher, *Deep learning.* MIT press, 2019.

[25] Y.-T. Kim, "Contrast enhancement using brightness preserving bi-histogram equalization," *IEEE transactions on Consumer Electronics*, vol. 43, no. 1, pp. 1–8, 1997.

[26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[27] M. Matsugu, K. Mori, Y. Mitari, and Y. Kaneda, "Subject independent facial expression recognition with robust face detection using a convolutional neural network," *Neural Networks*, vol. 16, no. 5-6, pp. 555–559, 2003.

[28] S. Qian, H. Liu, C. Liu, S. Wu, and H. San Wong, "Adaptive activation functions in convolutional neural networks," *Neurocomputing*, vol. 272, pp. 204–212, 2018.

[29] B. B. Shabarinath and P. Muralidhar, "Convolutional Neural Network based Traffic-Sign Classifier Optimized for Edge Inference," in *2020 IEEE REGION 10 CONFERENCE (TENCON)*, Nov. 2020, pp. 420–425, iSSN: 2159-3450.

[30] L. Shangzheng, "A traffic sign image recognition and classification approach based on convolutional neural network," in *2019 11th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*. IEEE, 2019, pp. 408–411.

[31] S. Sharma and S. Sharma, "Activation functions in neural networks," *Towards Data Science*, vol. 6, no. 12, pp. 310–316, 2017.

[32] P. Shopa, N. Sumitha, and P. S. K. Patra, "Traffic sign detection and recognition using OpenCV," in *International Conference on Information Communication and Embedded Systems (ICICES2014)*, Feb. 2014, pp. 1–6.

[33] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[34] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay," *arXiv preprint arXiv:1803.09820*, 2018.

[35] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[36] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, no. 0, pp. –, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608012000457

[37] T. S. Tsoi and C. Wheelus, "Traffic Signal Classification with Cost-Sensitive Deep Learning Models," in *2020 IEEE International Conference on Knowledge Graph (ICKG)*, Aug. 2020, pp. 586–592.

[38] F. Zaklouta and B. Stanciulescu, "Real-time traffic-sign recognition using tree classifiers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1507–1514, 2012.

Appendix A
Model code snippet

The following code snippet takes a pre-processed image and classifies it into a corresponding label.

```
def myModel():
    no_Of_Filters = 60
    size_of_Filter = (5, 5)
    size_of_Filter2 = (3, 3)
    size_of_pool = (2, 2)
    no_Of_Nodes = 500  # NO. OF NODES IN HIDDEN LAYERS
    model = Sequential()
    model.add((Conv2D(no_Of_Filters, size_of_Filter,

    input_shape=(imageDimesions[0], imageDimesions[1], 1),
                          activation='relu')))
    model.add((Conv2D(no_Of_Filters, size_of_Filter,

    activation='relu')))
    model.add(MaxPooling2D(pool_size=size_of_pool))

    model.add((Conv2D(no_Of_Filters // 2, size_of_Filter2,

    activation='relu')))
    model.add((Conv2D(no_Of_Filters // 2, size_of_Filter2,

    activation='relu')))
    model.add(MaxPooling2D(pool_size=size_of_pool))
    model.add(Dropout(0.5))

    model.add(Flatten())
    model.add(Dense(no_Of_Nodes, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(noOfClasses, activation='softmax'))
    # COMPILE MODEL
    model.compile(Adam(lr=0.001), loss='categorical_crossentropy',
    metrics=['accuracy'])
    return model
```