Project Phase-1 Report

on

# TRAFFIC SIGN CLASSIFICATION

*Submitted to*

**NMAM INSTITUTE OF TECHNOLOGY, NITTE**

(An Autonomous Institution Under VTU, Belagavi)

*In partial fulfilment of the requirements for the award of the*

Degree of Bachelor of Engineering

In

Computer Science and Engineering

*By*

**Ankith Bhandary 4NM17CS020**

**Dhruv Shetty 4NM17CS057**

**KISHAN 4NM17CS090**

**MANUKASHYAP U V 4NM17CS101**

Under the guidance of

**Mr.** **RAMESHA SHETTIGAR**

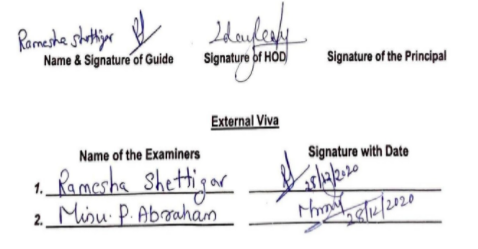
**Asst. Professor**

Dept. of CSE, NMAMIT, NITTE



CERTIFICATE

Certified that the project work entitled **Traffic Sing Classification** is a bonafide work carried out by Ankith Bhandary (4NM17CS020), Dhruv Shetty (4NM17CS057), Kishan (4NM17CS090), Manukashyap U V (4NM17CS101) in partial fulfillment for the award of Degree of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi during the year 2020-2021.It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project Phase - 2 prescribed for the said Degree.



# ACKNOWLEDGEMENT

The satisfaction that accompanies the completion of any task would be incomplete without the mention of all the people, without whom this endeavour would have been a difficult one to achieve. Their constant blessings, encouragement, guidance and suggestions have been a constant source of inspiration.

First and foremost, our gratitude to our project guide, **Mr. Ramesha Shettigar** for his constant guidance throughout the course of this project Phase - 2 and for the valuable suggestions.

We also take this opportunity to express a deep sense of gratitude to the project coordinators for their valuable guidance and support.

We acknowledge the support and valuable inputs given by, **Dr. Uday Kumar Reddy** the Head of the Department, Computer Science and Engineering, NMAMIT, Nitte.

Our sincere thanks to our beloved principal, **Dr. Niranjan N Chiplunkar** for permitting us to carry out this project at our college and providing us with all the needed facilities.

Finally, thanks to staff members of the Department of Computer Science and Engineering and our friends for their honest opinions and suggestions throughout the course of our project Phase - 2.

Ankith Bhandary (4NM17CS020)

Dhruv Shetty (4NM17CS057)

Kishan (4NM17CS090)

Manukashyap U V (4NM17CS101)

# ABSTRACT

With the advent of electric vehicles and drive by wire, the idea of adding autonomy to motor vehicles became mainstream. Even though traffic signs have been around for ages now, people tend to ignore most of them, of which the speed limit is the most prominent. If we can create a machine learning model that will detect and classify the traffic signs, the ECU can take over, under the scenario where the driver ignores the signs.

With this project we aim to create a machine learning model that is trained over a bench mark set of traffic signs and be able to detect most of the traffic signs with most accuracy. We also aim to create a camera interface to this model that can recognise the traffic signs in front of it in close to real time.

Table of Contents

[TRAFFIC SIGN CLASSIFICATION 1](#_Toc69305649)

[CERTIFICATE 2](#_Toc69305650)

[ACKNOWLEDGEMENT 3](#_Toc69305651)

[ABSTRACT 4](#_Toc69305652)

[INTRODUCTION 6](#_Toc69305653)

[Computer Vision 6](#_Toc69305654)

[Convolution Neural Network 6](#_Toc69305655)

[Computer Vision in Automobiles 6](#_Toc69305656)

[Traffic Sign Classification 6](#_Toc69305657)

[LITERATURE SURVEY 7](#_Toc69305658)

[**Paper - 1** 7](#_Toc69305659)

[**Paper - 2** 8](#_Toc69305660)

[**Paper - 3** 9](#_Toc69305661)

[**Paper - 4** 10](#_Toc69305662)

[**Paper - 5** 11](#_Toc69305663)

[PROBLEM DEFINITION 12](#_Toc69305664)

[Problem Statement 12](#_Toc69305665)

[Definition 12](#_Toc69305666)

[SYSTEM REQUIREMENTS SPECIFICATION 13](#_Toc69305667)

[Model Creation 13](#_Toc69305668)

[Model Execution 13](#_Toc69305669)

[Python Libraries 13](#_Toc69305670)

[SYSTEM DESIGN 14](#_Toc69305671)

[IMPLEMENTATION 15](#_Toc69305672)

[Importing the required libraries 15](#_Toc69305673)

[Setting up the parameters 15](#_Toc69305674)

[Importing the images 15](#_Toc69305675)

[Splitting the data 16](#_Toc69305676)

[Dataset Validation 16](#_Toc69305677)

[Reading the label file and displaying sample images 17](#_Toc69305678)

[Display information about the data set 18](#_Toc69305679)

[Pre-processing the images 18](#_Toc69305680)

[Image Augmentation 19](#_Toc69305681)

[Creating the Convolution Neural Network 20](#_Toc69305682)

[Training the model 20](#_Toc69305683)

[Plotting the Loss and Variation Graph 21](#_Toc69305684)

[Saving the model 21](#_Toc69305685)

[RESULTS AND CONCLUSION 22](#_Toc69305686)

[Phase - 1 Results and Conclusion 22](#_Toc69305687)

[Phase – 2 Results and Conclusion 22](#_Toc69305688)

[FUTURE WORKS 23](#_Toc69305689)

[Phase – 1 Future Works: 23](#_Toc69305690)

[Phase – 2 Future Works: 23](#_Toc69305691)

[REFERENCES 24](#_Toc69305692)

CHAPTER - 1

# INTRODUCTION

## Computer Vision

Definition: “Computer vision is an interdisciplinary scientific field that deals with how computers can gain a high-level understanding from digital images or videos. From the perspective of engineering, it seeks to understand and automate tasks that the human visual system can do.”

Computer vision enables the computer to partially interact with the real world and gain valuable data over time that will enable next level of automation in the day-to-day tasks leading to a safer and comfortable world for humans while also decreasing the global energy footprint.

## Convolution Neural Network

Definition: “In deep learning, a convolutional neural network is a class of deep neural networks, most commonly applied to analysing visual imagery. They are also known as shift invariant or space invariant artificial neural networks, based on their shared-weights architecture and translation invariance characteristics.”

Neural networks try to mimic the behaviour of the human brain to learn from real-world data and to solve real-world problems in the process. Convolution neural networks are one of the more efficient ways of using machine learning on images since they are capable of differentiating and identifying the different characters in each image, thereby accurately predicting the image class.

## Computer Vision in Automobiles

Lately, the use of computer vision in the field has skyrocketed. Computer vision provides unprecedented security and comfort for the automobile and its occupants. Some of these services are collision detention, traffic sign detection, driver awareness detection, side collision detection, traffic awareness etc. But these features require a huge amount of Research and Development (R&D) budget and also highly capable processing units to be installed on the vehicles. Thus, these features are only available in luxury vehicles whose cost would be the entire lifetime earnings of a common man a century ago.

## Traffic Sign Classification

Traffic sign classification is the process of automatically recognizing traffic signs along the road, including speed limit signs, yield signs, merge signs etc. Being able to automatically recognize the traffic sign enables us to build smarter and safer automobiles.

CHAPTER - 2

# LITERATURE SURVEY

## **Paper - 1**

**Paper Name:** **Towards Reliable Traffic Sign Recognition**

-Benjamin H ̈oferlin

Intelligent Systems Group Universit ̈ at Stuttgart, Germany Benjamin. höferlin@vis.uni-stuttgart.de

-Klaus Zimmermann

European Technology Centre (EuTEC) Sony Deutschland GmbH Stuttgart, Germany klaus.zimmermann@sony.de

**Abstract:**

* System architecture for the reliable detection of circular traffic signs.
* 80s first research into computer-aided traffic sign detection.
* How old research is using colour segmentation, colour thresholding, Bayesian classification of colour.
* Hough transform and its derivatives.

**Proposed Method:**

* Detection

○ Twofold detection stage

■ Shape-based detection

■ Content-based detection

* Refinement

○ Contracting Curve Density Algorithm

* Classification

○ Two Multi-Layered Perceptron

**Conclusion:**

* The 30-minute-long test yielded 96.4% correct detections.

**Future Work:**

* Detection of non-circular signs
* Better methods for classification [1].

## **Paper - 2**

**Paper Name:** **Detection and Recognition of Indian Traffic Signs**

- Pritika Priya, Dhara Modha, Mansi Agrawal Department Of Information Technology, Bharati Vidyapeeth College Of Engineering For Women, Pune-43

**Abstract:**

* An automatic system which would detect, recognize and interpret the meaning of the traffic signs for the driver.
* Use of several image processing techniques to enhance the efficiency and speed of the system.
* Disburden the drivers and reduce road accidents for better and safe driving, hence implementing the concept of the intelligent vehicle.

**Proposed Method:**

* Detection

○ Image blurring algorithm

○ Colour filtering

○ Blob detection

* Classification

○ Based on shapes: circle, rectangle and triangle.

* Recognition

○ Use of pattern matching algorithms to compare extracted ROI with

standard templates.

○ If a pattern is found, sound notification is given to the driver otherwise the

image is discarded.

**Conclusion:**

* Describes the system that is strictly used to differentiate Indian Traffic Signs that is subdivided into three classes according to the shapes.

**Future Work:**

* Improve the robustness of the system.
* Make the system work for highly tilted signs [2].

## **Paper - 3**

**Paper Name:** **Deep Learning for Large-Scale Traffic-Sign Detection and Recognition**

-Domen Tabernik and Danijel Skoˇcaj Faculty Computer and Information Science, University of Ljubljana Veˇcna pot 113,1000 Ljubljana {domen.tabernik,danijel.skocaj}@fri.uni-lj.si

**Abstract:**

* Use of Mask R-CNN for Traffic sign detection and recognition.
* Using deep learning method for detection of traffic signs with large intra-category appearance variation.
* This approach is used for the detection of 200 traffic-sign categories.

**Proposed Method:**

* Detection and Recognition using Mask R-CNN

○ Online hard-example mining (OEHM)

○ Distribution of selected training sample

○ Sample weighting

○ Adjusting region pass-through during detection

* Data augmentation technique

○This technique is used to generate several instances of traffic-signs and

hence provide diverse data for the deep learning model.

**Conclusion:**

* Describes the system that is strictly used to differentiate Indian Traffic Signs that is subdivided in three classes according to the shapes.

**Future Work:**

* Average precision of 97.5% achieved in correct detections with an error rate of just 2%-3%.
* Improving the system to achieve ideal performance [3].

## **Paper - 4**

**Paper Name:** **Traffic Sign Classification Using Deep Inception Based**

**Convolutional Networks**

-Mrinal Haloi, IIT Guwahati

mrinal.haloi11@gmail.com

**Abstract:**

* Use of spatial transformer layers and a modified version of inception module

specifically designed for capturing local and global features together.

* Classify precisely intraclass samples even under deformations.
* This approach addresses the concern of exploding parameters and

augmentations.

**Proposed Method:**

* Transformation invariant

○ Localization network

○ Grid generator

○ Sampling unit

● Proposed Pipeline

○ A modified version of GoogLeNet Inception module is used for the

classification task.

* GTSRB data set is used for training and testing.

**Conclusion:**

* Achieves the state-of-the-art performance of 99.81% on GTSRB dataset.

**Future Work:**

* Improving the system to achieve ideal performance [4].

## 

## **Paper - 5**

**Paper Name:** **Indian Traffic Sign Detection and Classification Using Neural**

**Network**

-Arun Nandewal, CSE Department, arunnandewal@gmail.com

-Abhishek Tripathi, IT Department, abhishek.tripathi2421@gmail.com

-Satyam Chandra, EEE Department, satyam9871@gmail.com

NITK Surathkal

**Abstract:**

* This paper presents an automatic Indian Road Traffic Sign Detection and Classification system based on Multiple Neural Networks.
* Validated on a standard data set of Indian Traffic Signs.
* The proposed methodology works with real-time images invariant to rotation, illumination and partially distorted and occluded images.

**Proposed Method:**

* The proposed system has 4 stages:

○ Image procurement and pre-processing

○ Colour segmentation

○ Blob Detection using Binarization and Otsu Thresholding.

○ Classification using Multiple Neural Networks to decide the

type of sign.

**Conclusion:**

* When the NN is trained over a standard database, the recognition of ROI has high accuracy.

**Future Work:**

* Real-time implementation requires a more robust system which has reduced

proceeding time [5].

CHAPTER - 3

# PROBLEM DEFINITION

## Problem Statement

Creation of an automatic, fast and light system that is capable of real-time detection, classification and interpretation of the traffic signs.

## Definition

Traffic sign classification is one of the most important uses of computer vision in automobiles. It has several aspects that include differentiating the road and surroundings, identifying the sign poles, closing in on the traffic sign and classifying it. We plan to tackle the issue of classifying the traffic signs once we have the sign in the view of the camera since it is the basic block of a traffic sign classification system.

Our goal is to create a machine learning model that is capable of classifying most of the traffic signals and also enable real-time interpretation of those signals. We aim to create an end product that can see the traffic signal through the camera and can interpret it onto the display while also being light on its resources.[8]

We plan to use OpenCV for the image input part of the system and the CNN model for the classification and interpretation of the traffic signs.[6]

CHAPTER - 4

# SYSTEM REQUIREMENTS SPECIFICATION

## Model Creation

* Any one of the following
  + Google Collab Notebook with GPU acceleration enabled.
  + Kaggle Notebook with GPU acceleration and Network enabled.
  + PC or Laptop with
    - Intel i5 or Ryzen 5 1600 or higher.
    - Nvidia GTX or RTX or Radeon series graphics card.
    - 1GB of storage space.
    - 8GB or more RAM
    - Anaconda or PyCharm IDE installed

## Model Execution

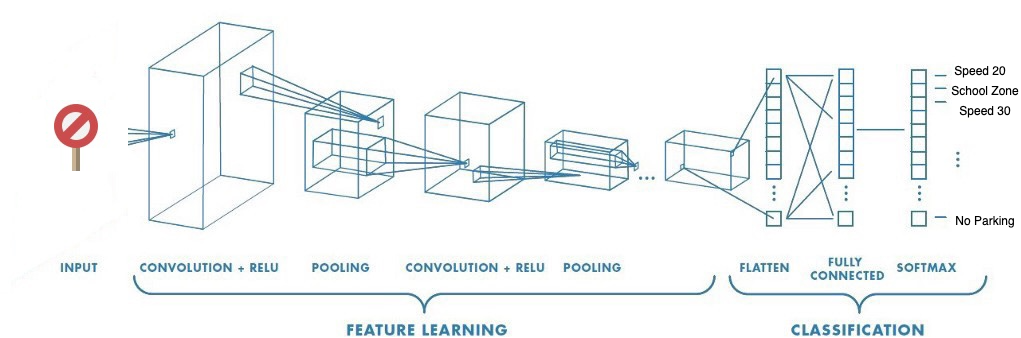
* Average PC or Laptop with Webcam.

## Python Libraries

* Python libraries such as NumPy, Matplotlib, TensorFlow, Keras, Cv2, Sklearn etc are used.

CHAPTER - 5

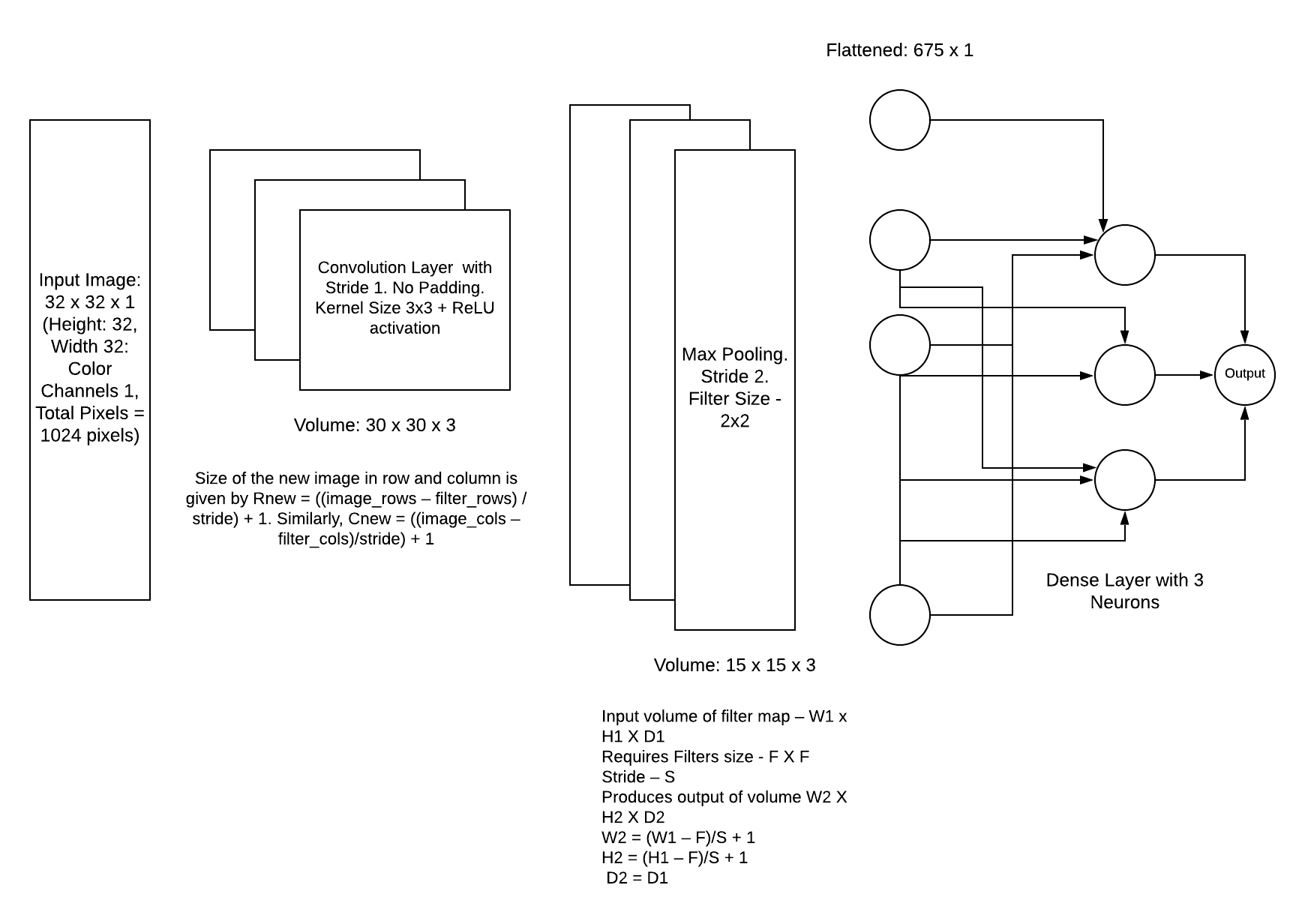
# SYSTEM DESIGN



CNN is a machine learning algorithm that can take an input image, assign importance to various aspects or objects in the image, and be able to differentiate one from another.[7]

[6]CNN works by extracting the features from the images. Any CNN consists of the following:

1. The input layer which is a grayscale image.
2. The output layer is a binary or multi-class label.
3. Hidden layers consist of convolution layers, ReLU (Rectified Linear Unit) layers, the pooling layer, and a fully connected Neural Network.



CHAPTER - 6

# IMPLEMENTATION

## Importing the required libraries

import NumPy as np

import matplotlib.pyplot as plt

from keras.models import Sequential

from keras.layers import Dense

from keras.optimizers import Adam

from keras.utils.np\_utils import to\_categorical

from keras.layers import Dropout, Flatten

from keras.layers.convolutional import Conv2D, MaxPooling2D

import cv2

from sklearn.model\_selection import train\_test\_split

import pickle

import os

import pandas as pd

import random

from keras.preprocessing.image import ImageDataGenerator

## Setting up the parameters

Different parameters such as the path for the dataset, batch size, epochs, steps per epoch etc are set up.

data\_path = "../input/tsc-data-set/myData/myData" # folder with all the class folders

label\_file = "../input/tsc-data-set/labels.csv" # file with all names of classes

batch\_size = 50 # how many to process together

steps\_per\_epoch = 400

epochs = 50

image\_dimensions = (32, 32, 3)

test\_ratio = 0.2 # if 1000 images split will 200 for testing

validation\_ratio = 0.2 # if 1000 images 20% of remaining 800 will be 160 for validation

## Importing the images

The images are imported from the dataset and loaded into the kernel for further processing.

count = 0

images = []

classNo = []

myList = os.listdir(data\_path)

print("Total Classes Detected:", len(myList))

noOfClasses = len(myList)

print("Importing Classes.....")

for x in range(0, len(myList)):

myPicList = os.listdir(data\_path+"/"+str(count))

for y in myPicList:

curImg = cv2.imread(data\_path+"/"+str(count)+"/"+y)

images.append(curImg)

classNo.append(count)

print(count, end=" ")

count += 1

print(" ")

images = np.array(images)

classNo = np.array(classNo)

Output

Total Classes Detected: 43

Importing Classes.....

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42

## Splitting the data

The dataset is split into test, train and validation sets.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

images, classNo, test\_size=test\_ratio)

X\_train, X\_validation, y\_train, y\_validation = train\_test\_split(

X\_train, y\_train, test\_size=validation\_ratio)

## Dataset Validation

The dataset and the label files are compared to check if there are any mismatches in the dataset.

print("Data Shapes")

print("Train", end="")

print(X\_train.shape, y\_train.shape)

print("Validation", end="")

print(X\_validation.shape, y\_validation.shape)

print("Test", end="")

print(X\_test.shape, y\_test.shape)

assert(X\_train.shape[0] == y\_train.shape[0]

), "The number of images in not equal to the number of lables in training set"

assert(X\_validation.shape[0] == y\_validation.shape[0]

), "The number of images in not equal to the number of lables in validation set"

assert(X\_test.shape[0] == y\_test.shape[0]

), "The number of images in not equal to the number of lables in test set"

assert(X\_train.shape[1:] == (image\_dimensions)

), " The dimesions of the Training images are wrong "

assert(X\_validation.shape[1:] == (image\_dimensions)

), " The dimesionas of the Validation images are wrong "

assert(X\_test.shape[1:] == (image\_dimensions)

), " The dimesionas of the Test images are wrong"

Output

Data Shapes

Train(22271, 32, 32, 3) (22271,)

Validation(5568, 32, 32, 3) (5568,)

Test(6960, 32, 32, 3) (6960,)

## Reading the label file and displaying sample images

Each label is read from the CSV file and corresponding images in the dataset is displayed for visual validation of the dataset.

# READ CSV FILE

data = pd.read\_csv(label\_file)

print("data shape ", data.shape, type(data))

# DISPLAY SOME SAMPLES IMAGES OF ALL THE CLASSES

num\_of\_samples = []

cols = 5

num\_classes = noOfClasses

fig, axs = plt.subplots(nrows=num\_classes, ncols=cols, figsize=(5, 100))

fig.tight\_layout()

for i in range(cols):

for j, row in data.iterrows():

x\_selected = X\_train[y\_train == j]

axs[j][i].imshow(x\_selected[random.randint(

0, len(x\_selected) - 1), :, :], cmap=plt.get\_cmap("gray"))

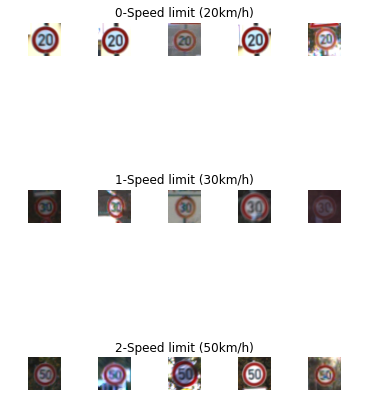
axs[j][i].axis("off")

if i == 2:

axs[j][i].set\_title(str(j) + "-"+row["Name"])

num\_of\_samples.append(len(x\_selected))

Output



## Display information about the data set

Some information about the distribution of images of different classes in the dataset is displayed.

# DISPLAY A BAR CHART SHOWING NO OF SAMPLES FOR EACH CATEGORY

print(num\_of\_samples)

plt.figure(figsize=(12, 4))

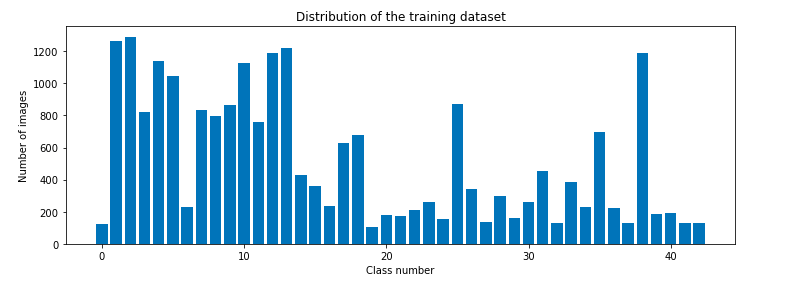
plt.bar(range(0, num\_classes), num\_of\_samples)

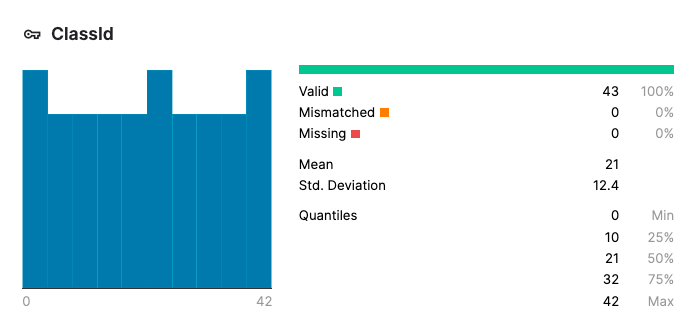
plt.title("Distribution of the training dataset")

plt.xlabel("Class number")

plt.ylabel("Number of images")

plt.show()





## Pre-processing the images

All of the images in the dataset is pre-processed to make all of them have the same dimensions and all of them are converted to grey scale images for easier processing.

# PREPROCESSING THE IMAGES

def grayscale(img):

img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

return img

def equalize(img):

img = cv2.equalizeHist(img)

return img

def preprocessing(img):

img = grayscale(img) # CONVERT TO GRAYSCALE

img = equalize(img) # STANDARDIZE THE LIGHTING IN AN IMAGE

img = img/255 # TO NORMALIZE VALUES BETWEEN 0 AND 1 INSTEAD OF 0 TO 255

return img

# TO IRETATE AND PREPROCESS ALL IMAGES

X\_train = np.array(list(map(preprocessing, X\_train)))

X\_validation = np.array(list(map(preprocessing, X\_validation)))

X\_test = np.array(list(map(preprocessing, X\_test)))

# ADD A DEPTH OF 1

X\_train = X\_train.reshape(

X\_train.shape[0], X\_train.shape[1], X\_train.shape[2], 1)

X\_validation = X\_validation.reshape(

X\_validation.shape[0], X\_validation.shape[1], X\_validation.shape[2], 1)

X\_test = X\_test.reshape(X\_test.shape[0], X\_test.shape[1], X\_test.shape[2], 1)

## Image Augmentation

# AUGMENTATAION OF IMAGES: TO MAKEIT MORE GENERIC

dataGen = ImageDataGenerator(width\_shift\_range=0.1, # 0.1 = 10% IF MORE THAN 1 E.G 10 THEN IT REFFERS TO NO. OF PIXELS EG 10 PIXELS

height\_shift\_range=0.1,

zoom\_range=0.2, # 0.2 MEANS CAN GO FROM 0.8 TO 1.2

shear\_range=0.1, # MAGNITUDE OF SHEAR ANGLE

rotation\_range=10) # DEGREES

dataGen.fit(X\_train)

# REQUESTING DATA GENRATOR TO GENERATE IMAGES BATCH SIZE = NO. OF IMAGES CREAED EACH TIME ITS CALLED

batches = dataGen.flow(X\_train, y\_train, batch\_size=20)

X\_batch, y\_batch = next(batches)

# TO SHOW AGMENTED IMAGE SAMPLES

fig, axs = plt.subplots(1, 15, figsize=(20, 5))

fig.tight\_layout()

for i in range(15):

axs[i].imshow(X\_batch[i].reshape(image\_dimensions[0], image\_dimensions[1]))

axs[i].axis('off')

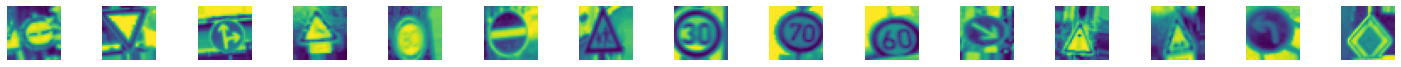
plt.show()

y\_train = to\_categorical(y\_train, noOfClasses)

y\_validation = to\_categorical(y\_validation, noOfClasses)

y\_test = to\_categorical(y\_test, noOfClasses)

Output



## Creating the Convolution Neural Network

After experimenting with different model parameters the most ideal CNN model is setup.

# CONVOLUTION NEURAL NETWORK MODEL

def myModel():

no\_Of\_Filters = 60

# THIS IS THE KERNEL THAT MOVE AROUND THE IMAGE TO GET THE FEATURES.

size\_of\_Filter = (5, 5)

# THIS WOULD REMOVE 2 PIXELS FROM EACH BORDER WHEN USING 32 32 IMAGE

size\_of\_Filter2 = (3, 3)

# SCALE DOWN ALL FEATURE MAP TO GERNALIZE MORE, TO REDUCE OVERFITTING

size\_of\_pool = (2, 2)

no\_Of\_Nodes = 500 # NO. OF NODES IN HIDDEN LAYERS

model = Sequential()

# ADDING MORE CONVOLUTION LAYERS = LESS FEATURES BUT CAN CAUSE ACCURACY TO INCREASE

model.add((Conv2D(no\_Of\_Filters, size\_of\_Filter, input\_shape=(

image\_dimensions[0], image\_dimensions[1], 1), activation='relu')))

model.add((Conv2D(no\_Of\_Filters, size\_of\_Filter, activation='relu')))

# DOES NOT EFFECT THE DEPTH/NO OF FILTERS

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'))

# INPUTS NODES TO DROP WITH EACH UPDATE 1 ALL 0 NONE

model.add(Dropout(0.5))

model.add(Dense(noOfClasses, activation='softmax')) # OUTPUT LAYER

# COMPILE MODEL

model.compile(Adam(lr=0.001), loss='categorical\_crossentropy',

metrics=['accuracy'])

return model

## Training the model

# TRAIN

model = myModel()

print(model.summary())

history = model.fit\_generator(dataGen.flow(X\_train, y\_train, batch\_size=batch\_size),

steps\_per\_epoch=steps\_per\_epoch, epochs=epochs, validation\_data=(X\_validation, y\_validation), shuffle=1)

## Plotting the Loss and Variation Graph

The graph of how the accuracy and loss varies with each of the epoch is plotted.

# PLOT

plt.figure(1)

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.legend(['training', 'validation'])

plt.title('Loss Variation Graph')

plt.xlabel('epoch')

plt.ylabel('Loss')

plt.figure(2)

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.legend(['training', 'validation'])

plt.title('Accuracy Variation Graph')

plt.xlabel('epoch')

plt.ylabel('Accuracy')

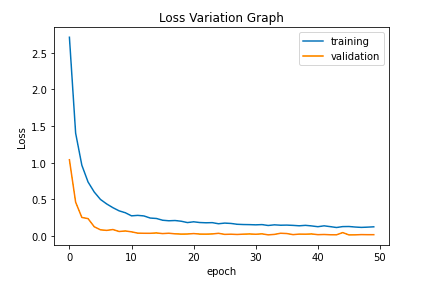
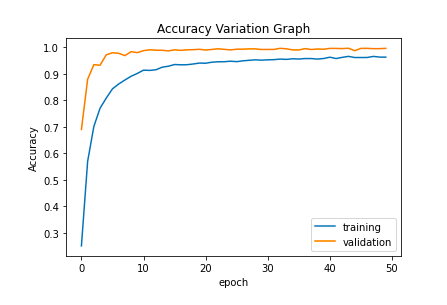
plt.show()

score = model.evaluate(X\_test, y\_test, verbose=0)

print('Test Score:', score[0])

print('Test Accuracy:', score[1])

Output



Test Score: 0.012194344773888588

Test Accuracy: 0.995976984500885

## Saving the model

The model is saved to the disk in the form of JSON and HDF5 for future operations.

# serialize model to JSON

model\_json = model.to\_json()

with open("model.json", "w") as json\_file:

json\_file.write(model\_json)

# serialize weights to HDF5

model.save\_weights("model.h5")

print("Saved model to disk")

CHAPTER - 7

# RESULTS AND CONCLUSION

## Phase - 1 Results and Conclusion

1. The data set does not have any irregularities or corrupt data.
2. The test train split passed the initial validation.
3. The distribution chart indicates sufficient images in all classes.
4. The colour space conversion of the images is successful.
5. The images are properly augmented and ready to be fed to the model.

## Phase – 2 Results and Conclusion

1. The model is created.
2. The model is trained.
3. The Loss and Accuracy Graphs are plotted.
4. The model is saved for future operations.

CHAPTER - 8

# FUTURE WORKS

## Phase – 1 Future Works:

1. Train the model
2. Test the model
3. Validate the model
4. Save the model
5. Create an interface to interact with the model
6. Add real-time interaction with the model using a webcam

## Phase – 2 Future Works:

1. Create a real time camera interface for the model.
2. Display the results in close to real time.

CHAPTER - 9

# REFERENCES

[1]. Towards Reliable Traffic Sign Recognition

Benjamin Hoferlin - Intelligent Systems Group Universitat Stuttgart Stuttgart, Germany benjamin.hoeferlin@vis.uni-stuttgart.de

Klaus Zimmermann - European Technology Center (ESTEC) Sony Deutschland GmbH Stuttgart, Germany [klaus.zimmermann@sony.de](mailto:klaus.zimmermann@sony.de)

[2]. Detection And Recognition Of Indian Traffic Signs

Pritika Priya, Dhara Modha, Mansi Agrawal Department Of Information Technology, Bharati Vidyapeeth College Of Engineering For Women, Pune-43

[3]. Deep Learning for Large-Scale Traffic-Sign Detection and Recognition

Domen Tabernik and Danijel Skocaj Faculty of Computer and Information Science, University of Ljubljana Vecna pot 113, 1000 Ljubljana {domen. tabernik,danijel.skocaj}@fri.uni-lj.si

[4]. Traffic Sign Classification Using Deep Inception Based Convolutional

Networks Mrinal Haloi IIT Guwahati1, [mrinal.haloi11@gmail.com](mailto:mrinal.haloi11@gmail.com)

[5]. Indian Traffic Sign Detection and Classification Using Neural Networks

Arun Nandewal - CSE Department, arunnandewal@gmail.com

Abhishek Tripathi - IT Department, abhishek.tripathi2421@gmail.com

Satyam Chandra - EEE Department, satyam9871@gmail.com

NITK Surathkal

[6]. Saha, S. (2018). A Comprehensive Guide to Convolutional Neural Networks

— the ELI5 way.

[7]. Stanford University (2018). Introduction to Convolutional Neural Networks.

[8]. Visin, V.D., Francesco (2016). English: Animation of a variation of the convolution operation. Blue maps are inputs, and cyan maps are outputs.

[online] Wikimedia Commons.

[9]. Bonner, A. (2019). The Complete Beginner’s Guide to Deep Learning:

Convolutional Neural Networks.