

TRAFFIC SIGN RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

Sanjay N M (38110499) & Sarvesh S (38110505)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC**

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI - 600 119

March - 2022



SATHYABAMA



INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with “A” grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600
119

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **SANJAY N M (38110499) & SARVESH S (38110505)** who carried out the project entitled “**CNN ALGORITHM TO DETECT TRAFFIC SIGNS**” under my supervision from **June 2021 to March 2022**.

Internal Guide

DR.S.DHAMODARAN ,Ph.D.,

Head of the Department

DR. VIGNESHWARI S, M.E., Ph. D.,

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

I, **Sanjay N M (38110499) & Sarvesh S (38110505)** hereby declare that the Project Report entitled “**CNN ALGORITHM TO DETECT TRAFFIC SIGNS**” done by me under the guidance of Dr S Dhamodaran Ph.D., ME (Internal) is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering / Technology degree in **Computer Science and Engineering**.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr.T.Sasikala M.E., Ph. D, Dean**, School of Computing, **Dr.S.Vigneshwari M.E., Ph.D.**, and **Dr.L.Lakshmanan M.E., Ph.D.**, Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.S.Dhamodaran.Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Traffic sign detection and recognition has always been a concern as it has many applications in the near future. One of the most important being autonomous cars etc. this project is to use convolutional neural network algorithms to train the model with a data set of different traffic signs, test and validate with a high accuracy rating. This is a basic step which is to come. It can be used for highway maintenance. driver support exercises and more. The given data set has 43 different classes. The model's accuracy for a given class is around 99(average) percent for signs that have a minimum of 500 images or more. The accuracy relatively drops to 80(avg) percent as the number of images drops to 100 or less. The reason for this is that the epochs(iterations) run are Ten. As the given data set is relatively huge the iterations take hours to train. The CNN algorithm is best for such images such as symbols as it trains a huge number of images before testing and putting in the grayscale feature makes it more efficient. This system can be introduced in people's daily life as it can save people from giving warnings to people about the traffic signs. This can be introduced in every vehicle just like the parking assistance that has been introduced in today's vehicles.

TABLE OF CONTENTS

Chapter No.	Title	Page No.
	Declaration	iii
	Acknowledgement	iv
	Abstract	v
	Table of Contents	vi
	List of Figures	vii
1	INTRODUCTION	01
	1.1 Outline	01
	1.2 Model IDE	01
	1.3 Problem Statement	01
	1.4 Objective	02
2	LITERATURE SURVEY	03
3	DEEP LEARNING	05
	3.1 Introduction to Deep Learning	05
	3.2 Neural Network	06
	3.3 Convolutional Neural Network	08
	3.4 Convolutional Layer -The Kernel	11
	3.5 Pooling Layer	13
	3.6 Fully Connected Layer	15
4	RESULTS AND DISCUSSION	19
	4.1 Results	19
	4.2 Analysis	23
	4.3 System Requirements	24
5	SUMMARY AND FUTURE SCOPE	26
	5.1 Summary	26
	5.2 Future Scope	26
	References	27
	APPENDIX - Source Code	29
	Screenshots	45

LIST OF FIGURES

S.No	Figures	Page No.
1	Programming Vs Machine Learning	5
2	Deep Learning Core	6
3	A Neuron	7
4	Process Outline	8
5	Image As Matrix	9
6	RGB Channel As Matrix	10
7	Kernel Layer For RGB	12
8	Types Of Pooling	14
9	Matrix In Formula.1	17
10	Matrix In Formula.2	17
11	Accuracy Graph	20
12	Confusion Matrix	21
13	Precision formula	22
14	Recall formula	22
15	F1-Score formula	23

CHAPTER – 1

INTRODUCTION

1.1. OUTLINE:

This project is used to traffic sign detection and recognition since it has received an increasing interest in the previous years and up to come. This is due to the wide range of applications that a system with this capability can provide driving assistance, highway maintenance, driver support systems and intelligent autonomous vehicles.

We use Convolutional Neural Network system to make the machine learn the traffic sign to use the given data set and learn the different traffic signs and live track the signs.

CNN is the best algorithm when it comes to tracking static images and has an accuracy above 90 percent.

1.2. MODEL IDE:

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The IDE was installed in Anaconda, an open-source distribution for the languages Python and R used to perform Data Science and Machine Learning. The IDE's UI in which the model was developed is given.

1.3. PROBLEM STATEMENT:

Traffic signs are an important part of road rules. This is a type of language that is spoken on roads. This is meant to read by people but is ignored by most people. Thus making the automobiles to read it by themselves

1.4. OBJECTIVE:

The usage of vehicles in every country has increased exponentially. Every person is starting to own a car of their own thus increasing the road traffic and more issues such as accidents.

As an example, in India 2020 there has been around 3.75 lakh accidental deaths according to NCRB data. This can be reduced significantly if people can follow the road signs and follow the rules accordingly.

Thus the given system uses computer vision to detect and recognize road signs with high accuracy. This is achieved by using python and deep learning, The algorithm used in particular is convolution neural network.

We use different libraries of python such as Tensor Flow and Open CV to achieve this. Thus reducing tragedies and creating new opportunities in this branch

CHAPTER – 2

LITERATURE SURVEY

In the years gone by, research on the topic of exploiting machine learning and deep learning algorithms to classify Traffic signs and advance them to the next steps of autonomous cars.

Liwei Jia, Xiaoming Shi

Traffic sign detection (TSD) and traffic sign classification (TSC) constitute a complete recognition system. The paper mainly studies the traffic sign recognition. Traffic sign recognition is mostly applied to portable devices, so the size and detection speed of the model are important factors to be considered. Under the condition of ensuring the speed, the detection accuracy of the model is guaranteed. (2020)

Template Matching is a high-level machine vision technique that identifies the parts on an image that match a predefined template. Advanced template matching algorithms allow to find occurrences of the template regardless of their orientation and local brightness.

It uses Template matching. the first disadvantage is that you need to know what you're looking for. template matching provided by OpenCV doesn't let you check for rotations and scaling

Sharifah Maryam Alhabshee, Abu Ubaidah Bin Shamsudin

In this paper, a deep learning method is used to make a system for traffic sign recognition. You Only Look Once (YOLOv3) is used as it has a quick response in terms of real-time data reliability followed by high accuracy and robust performance. This study applies image preprocessing for better decision making for the recognition system. (2020).

it's incredibly fast and can process 45 frames per second. YOLO also understands generalized object representation. This is one of the best algorithms for object detection and has shown a comparatively.

Struggles to detect close objects because each grid can propose only bounding boxers. Struggles to detect small objects. Due to high speed high error rates.

Zhilong Hei, Zhongjun Xiao

Traffic sign detection (TSD) and traffic sign classification (TSC) constitute a complete recognition system. The paper mainly studies the traffic sign recognition. Traffic sign recognition is mostly applied to portable devices, so the size and detection speed of the model are important factors to be considered. Under the condition of ensuring the speed, the detection accuracy of the model is guaranteed. (2020)

The accuracy of the model designed in this paper on the German traffic sign recognition benchmark (GTSRB) is 99.30%. The parameter size is only 1.3M, and the trained network model is 4.0M.

The number of classifiers used are less. This can lead to lot of clashes between similar signs

S Jayanthi Sree, A Surya Prakash

The proposed technique makes use of Gabor based kernel followed by a normal convolution kernel after the pooling layer. The optimizer technique used here is the Adams method. Hue, Saturation Value color space features have a speed of detection is faster and low suffering from illumination.. (2021).

The advantage of Gabor filters over Gaussian derivatives is related to a higher flexibility in the definition of the function shape, because of a more general set of features.

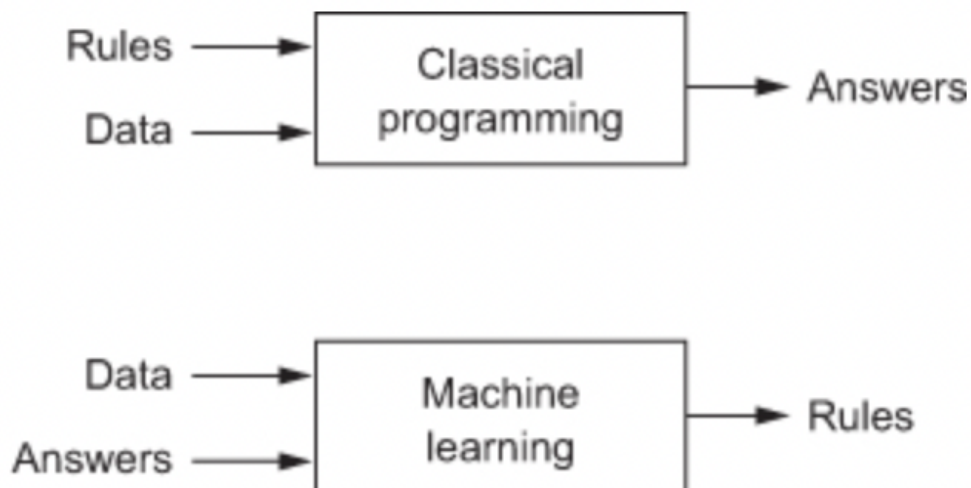
Challenge of Gabor technique is it has high redundancy of features. It uses RGB to detect classify symbols which causes more complications

CHAPTER – 3 DEEP LEARNING

3.1 INTRODUCTION TO DEEP LEARNING:

Deep learning is a subset of Machine Learning that uses mathematical functions to map the input to the output. These functions can extract non-redundant information or patterns from the data, which enables them to form a relationship between the input and the output.

This is known as learning, and the process of learning is called training. a field of computer science that uses statistical techniques to give computer systems the ability to "learn" (e.g., progressively improve performance on a specific task) with data, without being explicitly programmed.



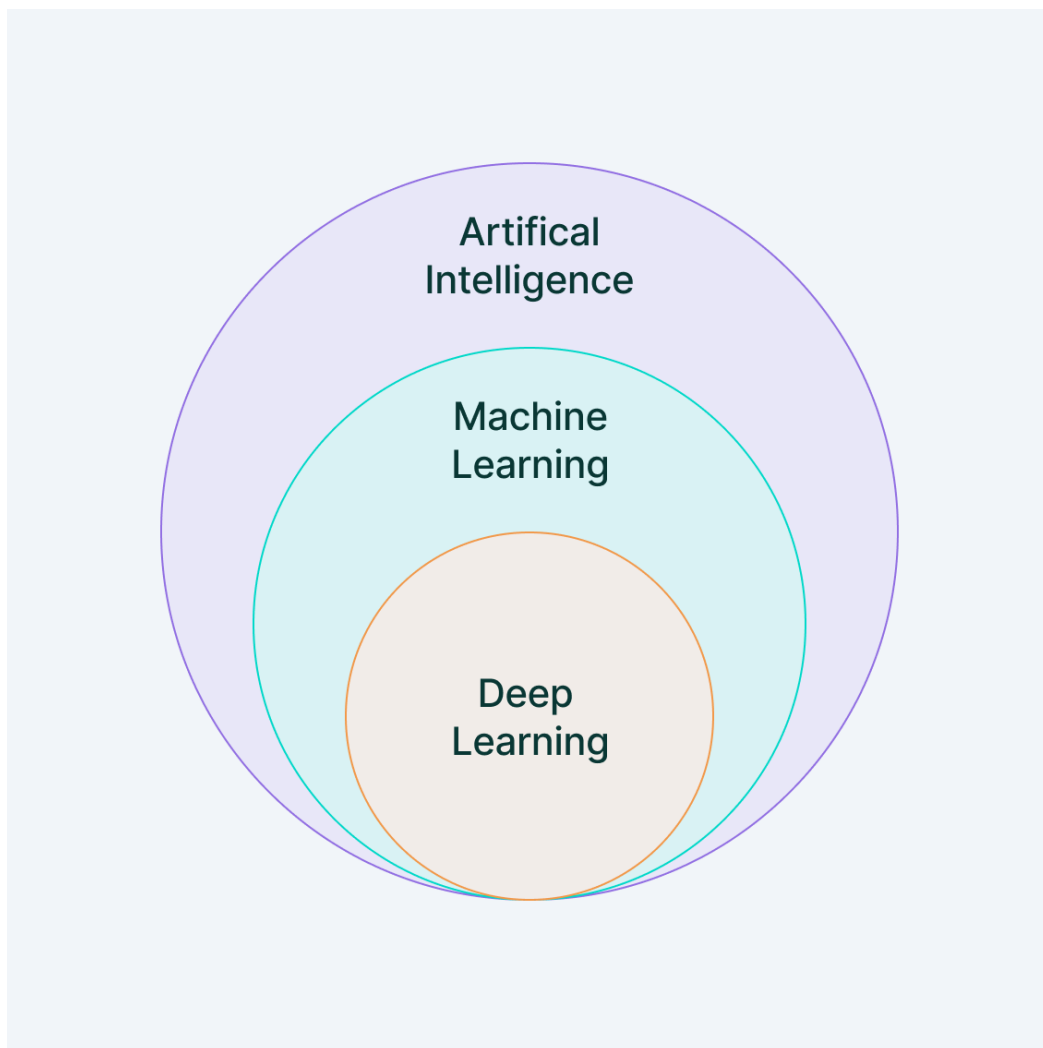
Programming Vs Machine Learning

In traditional computer programming, input and a set of rules are combined together to get the desired output. In machine learning and deep learning, input and output are correlated to the rules.

These rules—when combined with new input—yield desired results.

Modern deep learning models use artificial neural networks or simply neural networks to extract information.

These neural networks are made up of a simple mathematical function that can be stacked on top of each other and arranged in the form of layers, giving them a sense of depth, hence the term Deep Learning.



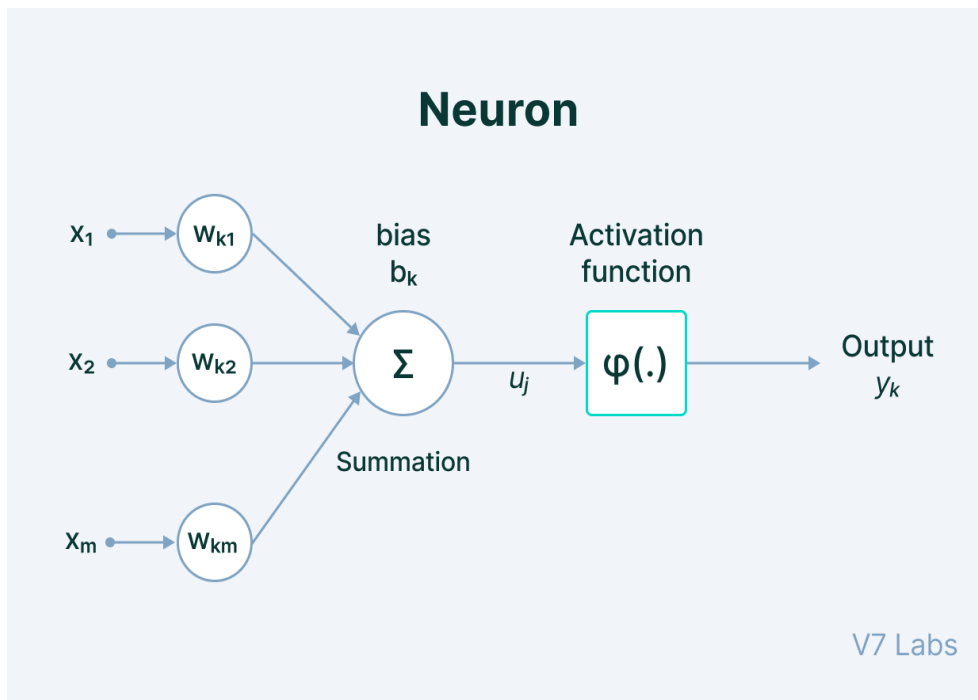
Deep learning Core

Deep learning can also be thought of as an approach to Artificial Intelligence, a smart combination of hardware and software to solve tasks requiring human intelligence.

3.2 NEURAL NETWORK:

The neural network is the heart of deep learning models, and it was initially designed to mimic the working of the neurons in the human brain.

Here are its components.



A Neuron

The neuronal perception of deep learning is generally motivated by two main ideas:

It is assumed that the human brain proves that intelligent behavior is possible, and—by reverse engineering, it is possible to build an intelligent system

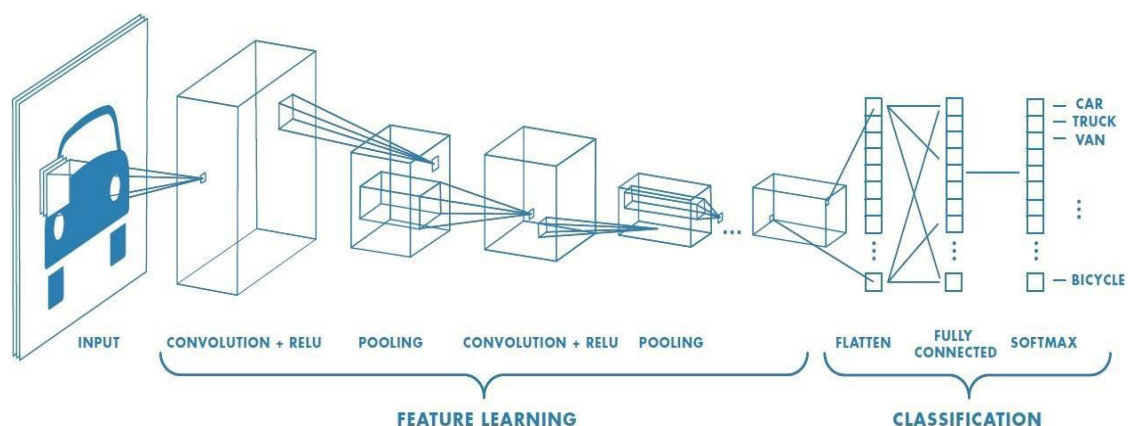
Another perspective is that to understand the working of the human brain and the principles that underlie its intelligence is to build a mathematical model that could shed light on the fundamental scientific questions.

In essence, neural networks enable us to learn the structure of the data or information and help us to understand it by performing tasks such as clustering, classification, regression, or sample generation.

3.3 CONVOLUTION NEURAL NETWORK:

Artificial Intelligence has been witnessing a monumental growth in bridging the gap between the capabilities of humans and machines. Researchers and enthusiasts alike, work on numerous aspects of the field to make amazing things happen. One of many such areas is the domain of Computer Vision.

The agenda for this field is to enable machines to view the world as humans do, perceive it in a similar manner and even use the knowledge for a multitude of tasks such as Image & Video recognition, Image Analysis & Classification, Media Recreation, Recommendation Systems, Natural Language Processing, etc. The advancements in Computer Vision with Deep Learning has been constructed and perfected with time, primarily over one particular algorithm — a Convolutional Neural Network.



Process Outline

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area

3.3.1 ConvNets over Feed-Forward Neural Nets:

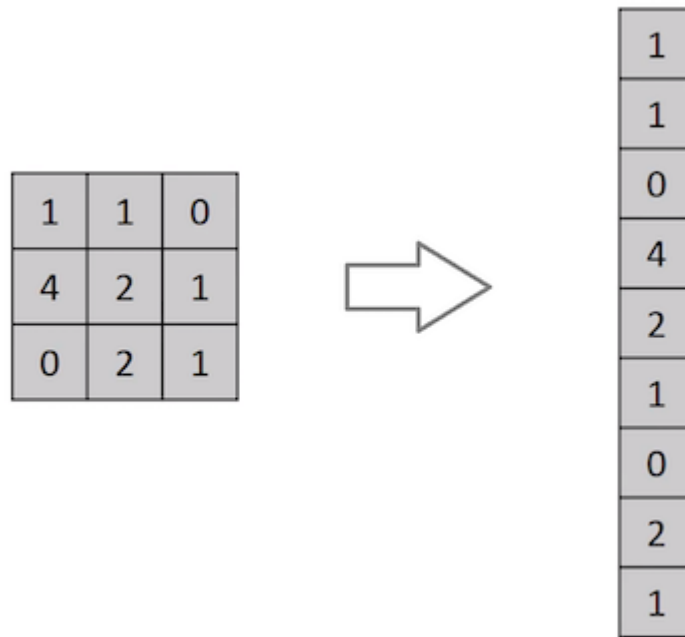


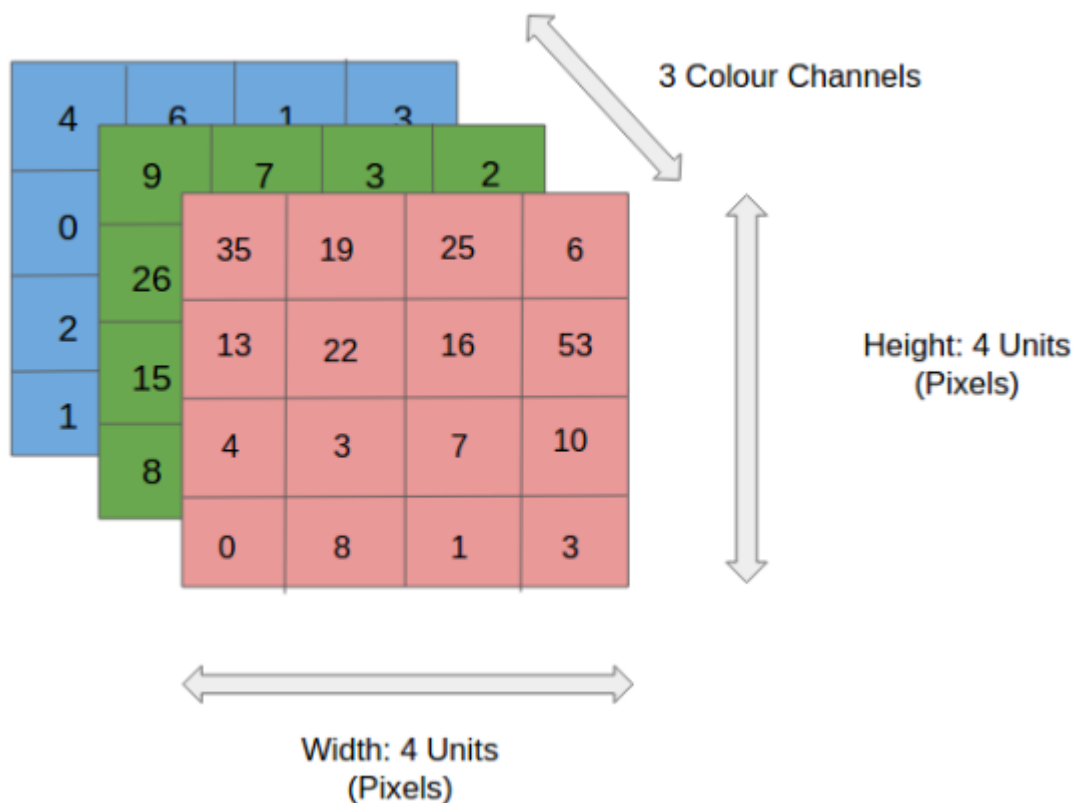
Image as Matrix

An image is nothing but a matrix of pixel values, right? So why not just flatten the image (e.g. 3x3 image matrix into a 9x1 vector) and feed it to a Multi-Level Perceptron for classification purposes? Uh.. not really.

In cases of extremely basic binary images, the method might show an average precision score while performing prediction of classes but would have little to no accuracy when it comes to complex images having pixel dependencies throughout.

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and re usability of weights. In other words, the network can be trained to understand the sophistication of the image better.

3.3.2 INPUT IMAGE:



RGB channel as a Matrix

In the figure, we have an RGB image which has been separated by its three color planes — Red, Green, and Blue. There are a number of such color spaces in which images exist — Grayscale, RGB, HSV, CMYK, etc.

You can imagine how computationally intensive things would get once the images reach dimensions, say 8K (7680×4320). The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

3.4 CONVOLUTION LAYER - THE KERNAL:

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved
Feature

Kernel Layer

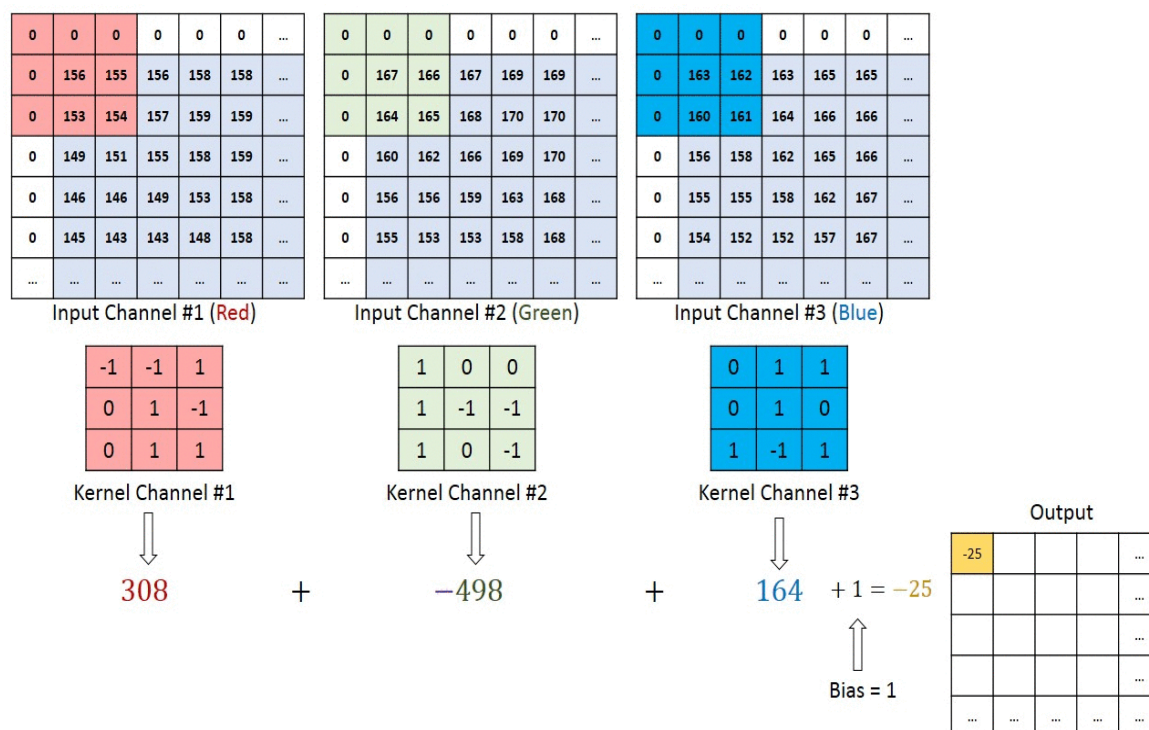
Image Dimensions = 5 (Height) x 5 (Breadth) x 1 (Number of channels, eg. RGB)

In the above demonstration, the green section resembles our 5x5x1 input image, I. The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the Kernel/Filter, K, represented in the color yellow. We have selected K as a 3x3x1 matrix.

```
Kernel/Filter, K = 1  0  1
0  1  0
1  0  1
```

The Kernel shifts 9 times because of Stride Length = 1 (Non-Strided), every time performing a matrix multiplication operation between K and the portion P of the image over which the kernel is hovering. The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire

image is traversed.



Kernel Layer for RGB

In the case of images with multiple channels (e.g. RGB), the Kernel has the same depth as that of the input image. Matrix Multiplication is performed between K_n and I_n stack ($[K_1, I_1]; [K_2, I_2]; [K_3, I_3]$) and all the results are summed with the bias to give us a squashed one-depth channel Convolved Feature Output.

The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network which has the wholesome understanding of images in the dataset, similar to how we would.

There are two types of results to the operation — one in which the convolved feature is reduced in dimensionality as compared to the input, and the other in which the dimensionality is either increased or remains the same. This is done by applying

Valid Padding in case of the former, or Same Padding in the case of the latter.

When we augment the $5 \times 5 \times 1$ image into a $6 \times 6 \times 1$ image and then apply the $3 \times 3 \times 1$ kernel over it, we find that the convolved matrix turns out to be of dimensions $5 \times 5 \times 1$. Hence the name — Same Padding.

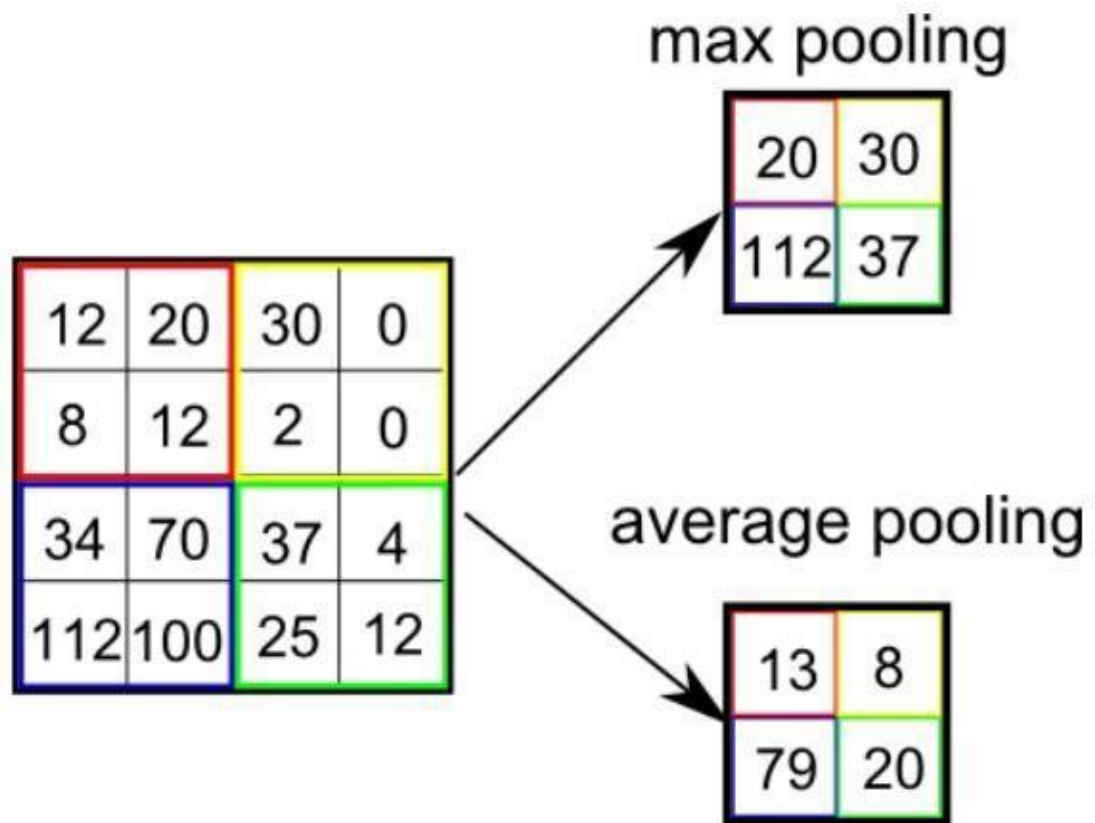
On the other hand, if we perform the same operation without padding, we are presented with a matrix which has dimensions of the Kernel ($3 \times 3 \times 1$) itself — Valid Padding.

3.5 Pooling Layer:

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model.

There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.

Max Pooling also performs as a Noise Suppressant. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that Max Pooling performs a lot better than Average Pooling.

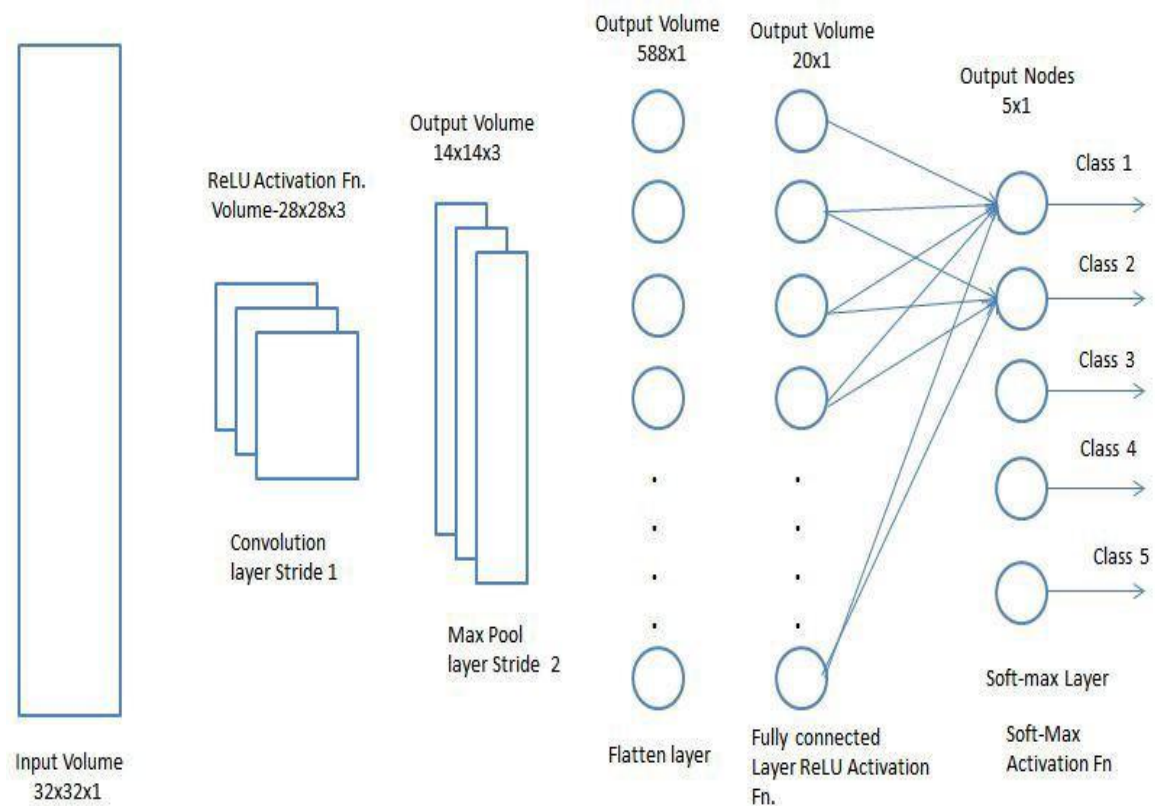


Types of Pooling

The Convolutional Layer and the Pooling Layer, together form the i -th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low-levels details even further, but at the cost of more computational power.

After going through the above process, we have successfully enabled the model to understand the features. Moving on, we are going to flatten the final output and feed it to a regular Neural Network for classification purposes.

3.6 Fully Connected Layer (FC Layer):



FC Layer

Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space.

Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every

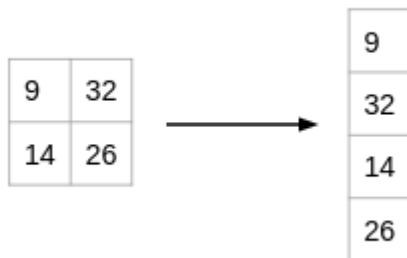
iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the Softmax Classification technique.

There are various architectures of CNNs available which have been key in building algorithms which power and shall power AI as a whole in the foreseeable future. Some of them have been listed below:

- LeNet
- AlexNet
- VGGNet
- GoogLeNet
- ResNet
- ZFNet

variable variables., the convolution layer has extracted some valuable features from the data. These features are sent to the fully connected layer that generates the final results. The fully connected layer in a CNN is nothing but the traditional neural network!

The output from the convolution layer was a 2D matrix. Ideally, we would want each row to represent a single input image. In fact, the fully connected layer can only work with 1D data. Hence, the values generated from the previous operation are first converted into a 1D format.



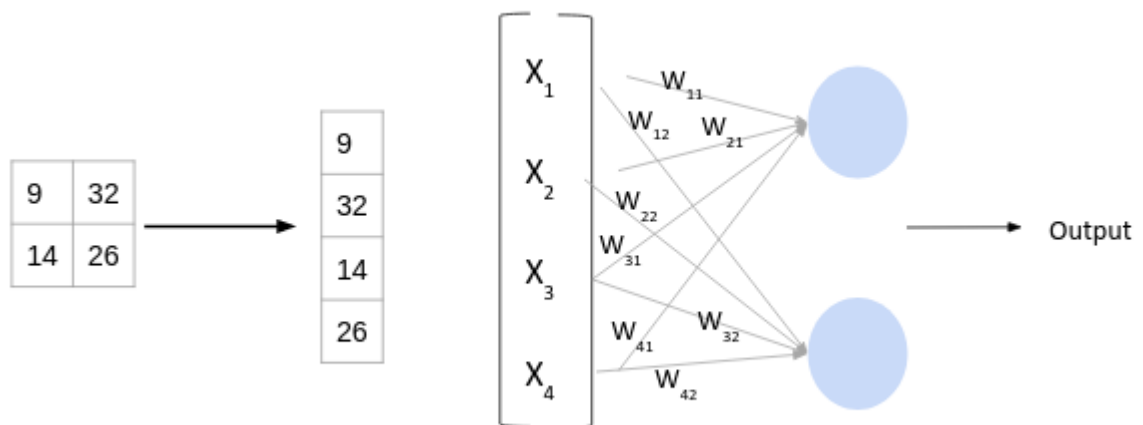
Example Matrix

Once the data is converted into a 1D array, it is sent to the fully connected layer. All of these individual values are treated as separate features that represent the image. The fully connected layer performs two operations on the incoming data – a linear transformation and a non-linear transformation.

We first perform a linear transformation on this data. The equation for linear transformation is:

$$Z = WT.X + b$$

Here, X is the input, W is weight, and b (called bias) is a constant. Note that the W in this case will be a matrix of (randomly initialized) numbers



Matrix In formula. 1

. In all cases, a function of the independent variables called the **regression function** is to be estimated.

Considering the size of the matrix is (m, n) – m will be equal to the number of features or inputs for this layer. Since we have 4 features from the convolution layer, m here would be 4. The value of n will depend on the number of neurons in the layer. For instance, if we have two neurons, then the shape of weight matrix will be $(4, 2)$:

$$X = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix}$$

Input Data

$$W = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \\ W_{31} & W_{32} \\ W_{41} & W_{42} \end{bmatrix}$$

Randomly Initialized
Weight Matrix

$$b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

Randomly Initialized
bias Matrix

Matrix in Formula. 2

Having defined the weight and bias matrix, let us put these in the equation for linear transformation:.

$$Z = W^T \cdot X + b$$

$$Z = \begin{bmatrix} W_{11} & W_{21} & W_{31} & W_{41} \\ W_{12} & W_{22} & W_{32} & W_{42} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

$$Z_{2 \times 2} = \begin{bmatrix} W_{11}X_1 + W_{21}X_2 + W_{31}X_3 + W_{41}X_4 \\ W_{12}X_1 + W_{22}X_2 + W_{32}X_3 + W_{42}X_4 \end{bmatrix}$$

Matrix In Formula. 3

CHAPTER – 4

RESULTS AND DISCUSSION

4.1 Results:

The result of the classification model prepared using Python may be divided into three parts in itself:

4.1.1 Pre-Model Data Visualization

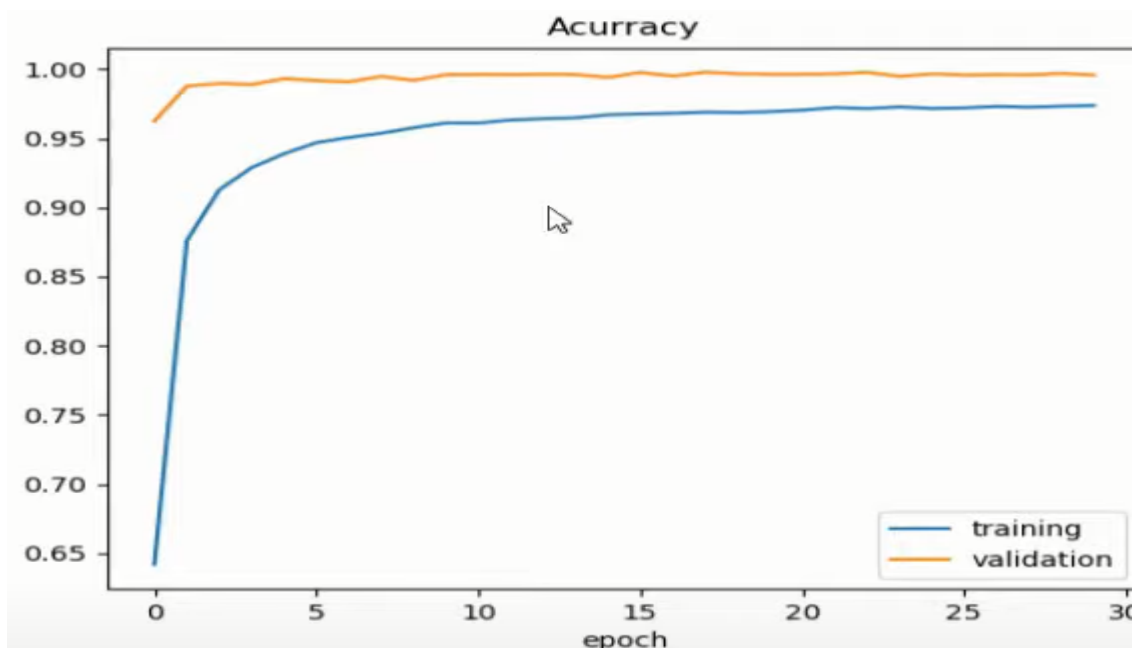
The pre-model data visualization section consists of all the data visualizations or example visualizations made with the pure data set to understand the features and

the spread of data in the data set.

Data Set:

It is traffic sign image dataset. The Traffic sign The Data set used in this CNN model is taken from kaggle. It has dimension of (39209,32,32,3) which means it has total of 39,209 images with 32x32 pixels set in the program, with 3 meaning RGB channels. The Dataset Has 43 classes starting from 0 to 42 each class representing a specific sign.

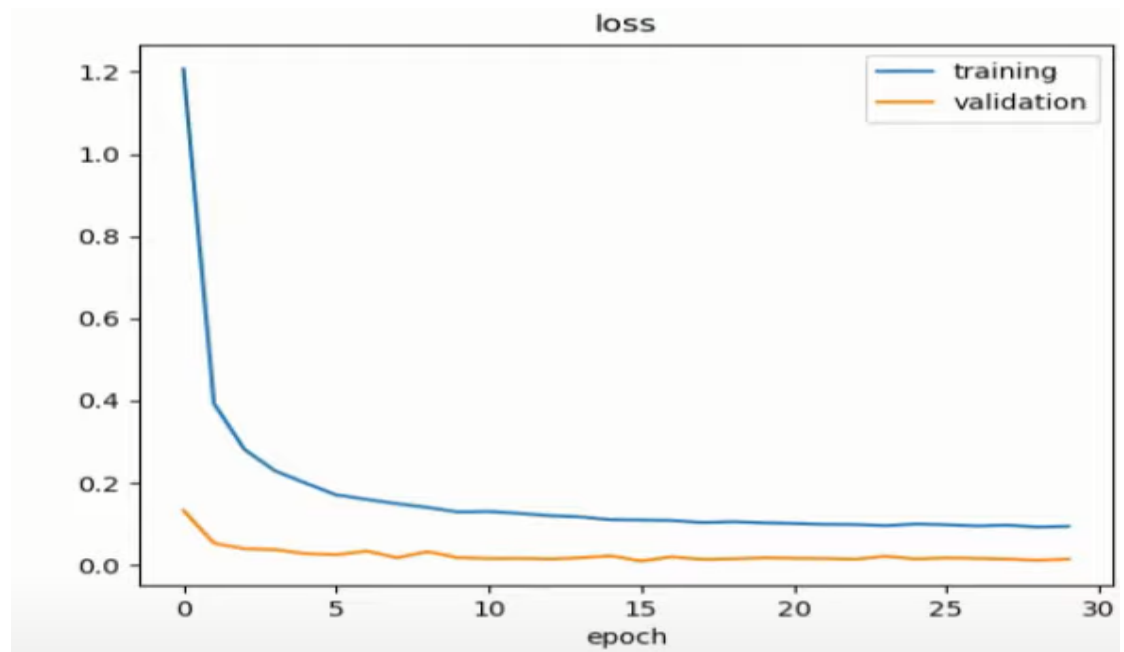
Visualizations:



Accuracy Graph

As shown in the graph the accuracy levels becomes stable after five epochs. Thus the model has achieved its highest accuracy and thus cannot move any further.

As for classes that have more number of images faster the accuracy is. In this model for classes that have more than 500 images have an accuracy rate above 95 percent and for classes that have only 200 images have an accuracy level of around 85 percent. The graph is pretty much opposite for the lose rate



Loss Graph

4.1.2 Model Information Visualization

In this part of the chapter, the visualizations with respect to the model will be discussed.

Confusion Matrix:



Precision:

$$\begin{aligned}\text{Precision} &= \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \\ &= \frac{\text{True Positive}}{\text{Total Predicted Positive}}\end{aligned}$$

Precision formula

Precision is defined as the fraction of the examples which are actually positive among all the examples which we predicted positive.

It is also defined as the number of true positives divided by the number of true positives plus the number of false positives. False positives are cases the model incorrectly labels as positive that are actually negative, or in our example, individuals the model classifies as terrorists that are not.

Recall:

$$\begin{aligned}\text{Recall} &= \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \\ &= \frac{\text{True Positive}}{\text{Total Actual Positive}}\end{aligned}$$

Recall formula

The metric our intuition tells us we should maximize is known in statistics as recall, or the ability of a model to find all the relevant cases within a dataset.

The precise definition of recall is the number of true positives divided by the number of true positives plus the number of false negatives. True positives are data point classified as positive by the model that actually are positive (meaning they are correct), and false negatives are data points the model identifies as negative that actually are positive (incorrect).

F1-Score:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

F-1 Score formula

Traditionally, it is defined to be the harmonic mean of precision and recall.

It is also called the F Score or the F Measure. Put another way, the F1 score conveys the balance between the precision and the recall.

4.2 ANALYSIS:

4.2.1 Research

There has been research done in this topic, but due to the decreased availability of real-time recognition or careless of human errors there are lot of accidents that are fatal. And with more development this model can be used to build in every car to automatically detect the errors and display the correct signs

4.2.2 Proposed System

The proposed system provides an accuracy of 98.4%, an Out of the Bag (OOB Score), used for CNN, of 0.99688 which is equivalent to 99.69% accuracy over model produced inputs and an R^2 score of 0.97963, which when nearer to 1 confirms of a stable model. The accuracy can be increased once reliable data with a more balanced dataset has been made.

The important features as predicted by the model are the Signs that the model finds to be the most influential in its decision making.

The proposed system uses the Jupyter Notebook IDE used in the Anaconda Environment for the building of the model itself and hence may require certain software requirements. The requirements for hardware is almost non-existent due to the fact that Jupyter Notebook runs on the internet. Hence, **Network Stability** plays a major role in the use of Jupyter Notebook.

4.3 SYSTEM REQUIREMENTS

4.3.1 System Requirements for Anaconda Free Distribution

- License: Free use and redistribution under the terms of the End User License Agreement.
- Operating system: Windows 7 or newer, 64-bit macOS 10.10+, or Linux, including Ubuntu, RedHat, CentOS 6+, and others.

If your operating system is older than what is currently supported, you can find older versions of the Anaconda installers in our archive that might work for you. Check our FAQ for version recommendations.

- System architecture: Windows- 64-bit x86, 32-bit x86; MacOS- 64-bit x86; Linux- 64-bit x86, 64-bit Power8/Power9.
- Minimum 5 GB disk space to download and install.

4.3.2 System Requirements for Tableau Desktop

Windows:

- Microsoft Windows 7 or newer (32-bit and 64-bit)
- Microsoft Server 2008 R2 or newer
- Intel Pentium 4 or AMD Opteron Processor or newer
- 2 GB memory
- 1.5 GB minimum free disk space

Mac:

- iMac/MacBook computers 2009 or newer
- macOS High Sierra 10.13, macOS Mojave 10.14 and macOS Catalina 10.15
- 1.5 GB minimum free disk

space Virtual Environments:

- Citrix environments, Microsoft Hyper-V, Parallels, VMware, Microsoft Azure and Amazon EC2.
- All of Tableau's products operate in virtualized environments when they are configured with the proper underlying Windows operating system and minimum hardware requirements.

APTER – 5
SUMMARY
AND FUTURE
SCOPE

5.1 SUMMARY:

This project was created to further create new opportunities for more autonomous and error free roads. Reading traffic signs as a language can help people to ride cars safely without any issues. Thus this model can be applied directly or can be upgraded for further finesse and be implied to solve most of the traffic issues in every country

5.2 FUTURE SCOPE:

I am thankful for this opportunity to work on this project. This project is one of my long thought research ideas which may turn to be useful. The future of this project has several opportunities.

This project, by my plan, is to be extended for further research with respect to my goal of establishing newer, more reliable and flexible methods of detecting traffic signs. This can also be used to not only detect traffic signs but every road rules. This can lead to creating a fully autonomous car that can be free of human errors and making the roads a safer place

This project can also be implemented for cops to navigate and read all the road accidents and record criminal activities that can help them investigate and get more clues.

And for new drivers this can be implemented for driver assistance programs and help them learn the road rules faster and drive safely. This can also be extend its use to maintenance systems on roads. It can solve a lot of issues on roads

References

1. Neural-Network-Based Traffic Sign Detection and Recognition in High-Definition Images Using Region Focusing and Parallelization ALEKSEJ AVRAMOVIĆ¹ , DAVOR SLUGA² , DOMEN TABERNIK², DANIJEL SKOČAJ² , (Member, IEEE), VLADAN STOJNIĆ¹ , (Member, IEEE), AND NEJC ILC² , (Member, IEEE) \Digital Object Identifier 10.1109/ACCESS.2020.3031191
2. Traffic Sign Recognition using Deep Learning for Autonomous Driverless Vehicles Vigneshwaran D ,Jayanthi Sree S,Suriya Prakash A ,Seenivasaga Ayyalu R, Suriya Prakash A DOI: 10.1109/ICCMC51019.2021.9418437
3. Real-time Traffic Signs Detection Based on YOLO Network Model. Wenkao Yang, Wei Zhang DOI:10.1109/CYBERC49757.2020.0006
4. DETECTION TECHNOLOGY OF TRAFFIC MARKING EDGE, Tingze Zhang, Huan Gao,DOI: 10.1109/ICCASIT50869.2020.9368865
5. Deep Learning Traffic Sign Recognition in Autonomous Vehicle, Sharifah Maryam Alhabshee, Abu Ubaidah bin Shamsudin;2020 IEEE Student Conference on Research and Development (SCOREd) 27-28 September 2020, Johor, Malaysia
6. Traffic Sign Recognition Based on Convolutional Neural Network Model,Zhulong, He, Zhongjun, Xiao,Zhiguo, Yan,DOI: 10.1109/CAC51589.2020.9327830
7. Sign of traffic sign detection and Recognition Algorithm based on template matching. Liwei Jia Xiaoming Shi Li Wei,DOI: 10.1109/ICCASIT50869.2020.9368750
8. Persian Text Based Traffic sign Detection with Convolutional Neural Network: A New Dataset Saba Kheirinejad,Noushin Riahi ,Reza Azmi 10th International Conference on Computer and Knowledge Engineering (ICCKE2020) October 29-30, 2020 Ferdowsi University of Mashhad - Iran SSLA Based Traffic Sign and Lane Detection for Autonomous cars, S Swetha, Dr.P. Sivakumar, DOI: 10.1109/ICAIS50930.2021.9396046
9. Detection of Malaysian Traffic Signs via Modified YOLOv3 Algorithm, Wan-Noorshahida Mohd-Isa,Mahmood Sarzil,Junaidi Abdullah,Aziah Ali,Noramiza Hashim DOI:10.1109/ICDABI51230.2020.9325690

10. Data set -

<https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign>

11. Anaconda Free Distribution - <https://www.anaconda.com/>

12. Jupyter Notebook - <https://jupyter.org/>

13. Data Science - https://en.wikipedia.org/wiki/Data_science

14. Machine Learning - https://en.wikipedia.org/wiki/Machine_learning

15. Scikit-learn - <https://scikit-learn.org/stable/>

16. Pandas - <https://pandas.pydata.org/>

17. Matplotlib - <https://matplotlib.org/>

18. Seaborn - <https://seaborn.pydata.org/>

19. CNN -

<https://www.analyticsvidhya.com/blog/2020/02/mathematics-behind-convolutional-neural-network/>

APPENDIX

SOURCE CODE

TRAINING CODE

```
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
```

Parameters

```
path = "myData" # folder with all the class folders
```

```
labelFile = 'labels.csv' # file with all names of classes
```

```
batch_size_val=50 # how many to process together
```

```
steps_per_epoch_val=2000
```

```
epochs_val=10
```

```
imageDimesions = (32,32,3)
```

```
testRatio = 0.2 # if 1000 images split will 200 for testing
```

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

Importing of the Images

```
count = 0
```

```
images = []
```

```
classNo = []
```

```
myList = os.listdir(path)
```

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

```
noOfClasses=len(myList)
```

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

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

```
    myPicList = os.listdir(path+"/"+str(count))
```

```
    for y in myPicList:
```

```
        curlmg = cv2.imread(path+"/"+str(count)+"/"+y)
```

```
        images.append(curlmg)
```

```
        classNo.append(count)
```

```
    print(count, end = " ")
```

```
    count +=1
```

```
print(" ")
```

```
images = np.array(images)
```

```
classNo = np.array(classNo)
```

Split Data

```
X_train, X_test, y_train, y_test = train_test_split(images, classNo, test_size=testRatio)
```

```
X_train, X_validation, y_train, y_validation = train_test_split(X_train, y_train, test_size=validationRatio)
```

```
# X_train = ARRAY OF IMAGES TO TRAIN
```

```
# y_train = CORRESPONDING CLASS ID
```

TO CHECK IF NUMBER OF IMAGES MATCHES TO NUMBER OF LABELS FOR EACH DATA SET

```
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:]==(imageDimesions)), " The dimesions of the Training images are wrong "
```

```
assert(X_validation.shape[1:]==(imageDimesions)), " The dimesionas of the Validation images are wrong "
```

```
assert(X_test.shape[1:]==(imageDimesions)), " The dimesionas of the Test images are wrong"
```

READ CSV FILE

```
data=pd.read_csv(labelFile)
```

```
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, 300))
```

```
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))
```

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()
```

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
```

```
X_train=np.array(list(map(preprocessing,X_train))) # TO ITERATE AND PREPROCESS ALL IMAGES
```

```
X_validation=np.array(list(map(preprocessing,X_validation)))
```

```
X_test=np.array(list(map(preprocessing,X_test)))
```

```
cv2.imshow("GrayScale Images",X_train[random.randint(0,len(X_train)-1)]) # TO CHECK IF THE  
TRAINING IS DONE PROPERLY
```

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)
```

AUGMENTATION OF IMAGES: TO MAKE IT MORE GENERIC

```
dataGen= ImageDataGenerator(width_shift_range=0.1, # 0.1 = 10%    IF MORE THAN 1 E.G 10  
THEN IT REFERS 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)

batches= dataGen.flow(X_train,y_train,batch_size=20) # REQUESTING DATA GENRATOR TO
GENERATE IMAGES BATCH SIZE = NO. OF IMAGES CREAED EACH TIME ITS CALLED

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(imageDimesions[0],imageDimesions[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)

```

CONVOLUTION NEURAL NETWORK MODEL

```
def myModel():
```

```
no_Of_Filters=60
```

```
size_of_Filter=(5,5) # THIS IS THE KERNEL THAT MOVE AROUND THE IMAGE TO GET THE  
FEATURES.
```

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

```
size_of_Filter2=(3,3)
```

```
size_of_pool=(2,2) # SCALE DOWN ALL FEATURE MAP TO GERNALIZE MORE, TO REDUCE  
OVERFITTING
```

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

```
model.add((Conv2D(no_Of_Filters, size_of_Filter, activation='relu')))
```

```
model.add(MaxPooling2D(pool_size=size_of_pool)) # DOES NOT EFFECT THE DEPTH/NO OF  
FILTERS
```

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

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

COMPILE MODEL

```
model.compile(Adam(lr=0.001),loss='categorical_crossentropy',metrics=['accuracy'])
```

```
return model
```

TRAIN

```
model = myModel()
```

```
print(model.summary())
```

```
history=model.fit_generator(dataGen.flow(X_train,y_train,batch_size=batch_size_val),steps_per_epoch  
=steps_per_epoch_val,epochs=epochs_val,validation_data=(X_validation,y_validation),shuffle=1)
```

PLOT

```
plt.figure(1)
```

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

```
plt.plot(history.history['val_loss'])
```

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

```
plt.title('loss')
```

```
plt.xlabel('epoch')
```

```
plt.figure(2)
```

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

```
plt.plot(history.history['val_accuracy'])
```

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

```
plt.title('Accuracy')
```

```
plt.xlabel('epoch')
```

```
plt.show()
```

```
score =model.evaluate(X_test,y_test,verbose=0)
```

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

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

```
# STORE THE MODEL AS A PICKLE OBJECT
```

```
pickle_out= open("model_trained.p","wb") # wb = WRITE BYTE
```

```
pickle.dump(model,pickle_out)
```

```
pickle_out.close()
```

```
cv2.waitKey(0)
```

TEST CODE

```
import numpy as np
```

```
import cv2
```

```
import pickle
```

```
frameWidth= 640      # CAMERA RESOLUTION
```

```
frameHeight = 480
```

```
brightness = 180
```

```
threshold = 0.75      # PROBABILITY THRESHOLD
```

```
font = cv2.FONT_HERSHEY_SIMPLEX
```

SETUP THE VIDEO CAMERA

```
cap = cv2.VideoCapture(0)
```

```
cap.set(3, frameWidth)
```

```
cap.set(4, frameHeight)
```

```
cap.set(10, brightness)
```

IMPORT THE TRANNIED MODEL

```
pickle_in=open("model_trained.p","rb") ## rb = READ BYTE
```

```
model=pickle.load(pickle_in)
```

```
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)
```

```
img = equalize(img)
```

```
img = img/255
```

```
return img
```

```
def getCalssName(classNo):
```

```
if classNo == 0: return 'Speed Limit 20 km/h'
```

```
elif classNo == 1: return 'Speed Limit 30 km/h'
```

```
elif classNo == 2: return 'Speed Limit 50 km/h'

elif classNo == 3: return 'Speed Limit 60 km/h'

elif classNo == 4: return 'Speed Limit 70 km/h'

elif classNo == 5: return 'Speed Limit 80 km/h'

elif classNo == 6: return 'End of Speed Limit 80 km/h'

elif classNo == 7: return 'Speed Limit 100 km/h'

elif classNo == 8: return 'Speed Limit 120 km/h'

elif classNo == 9: return 'No passing'

elif classNo == 10: return 'No passing for vechiles over 3.5 metric tons'

elif classNo == 11: return 'Right-of-way at the next intersection'

elif classNo == 12: return 'Priority road'

elif classNo == 13: return 'Yield'

elif classNo == 14: return 'Stop'

elif classNo == 15: return 'No vechiles'

elif classNo == 16: return 'Vechiles over 3.5 metric tons prohibited'

elif classNo == 17: return 'No entry'
```

elif classNo == 18: return 'General caution'

elif classNo == 19: return 'Dangerous curve to the left'

elif classNo == 20: return 'Dangerous curve to the right'

elif classNo == 21: return 'Double curve'

elif classNo == 22: return 'Bumpy road'

elif classNo == 23: return 'Slippery road'

elif classNo == 24: return 'Road narrows on the right'

elif classNo == 25: return 'Road work'

elif classNo == 26: return 'Traffic signals'

elif classNo == 27: return 'Pedestrians'

elif classNo == 28: return 'Children crossing'

elif classNo == 29: return 'Bicycles crossing'

elif classNo == 30: return 'Beware of ice/snow'

elif classNo == 31: return 'Wild animals crossing'

elif classNo == 32: return 'End of all speed and passing limits'

elif classNo == 33: return 'Turn right ahead'

```
elif classNo == 34: return 'Turn left ahead'
```

```
elif classNo == 35: return 'Ahead only'
```

```
elif classNo == 36: return 'Go straight or right'
```

```
elif classNo == 37: return 'Go straight or left'
```

```
elif classNo == 38: return 'Keep right'
```

```
elif classNo == 39: return 'Keep left'
```

```
elif classNo == 40: return 'Roundabout mandatory'
```

```
elif classNo == 41: return 'End of no passing'
```

```
elif classNo == 42: return 'End of no passing by vechiles over 3.5 metric tons'
```

```
while True:
```

```
# READ IMAGE
```

```
success, imgOriginal = cap.read()
```

```
# PROCESS IMAGE
```

```
img = np.asarray(imgOriginal)
```

```
img = cv2.resize(img, (32, 32))
```

```
img = preprocessing(img)
```

```
cv2.imshow("Processed Image", img)
```

```
img = img.reshape(1, 32, 32, 1)
```

```
cv2.putText(imgOriginal, "CLASS: ", (20, 35), font, 0.75, (0, 0, 255), 2, cv2.LINE_AA)
```

```
cv2.putText(imgOriginal, "PROBABILITY: ", (20, 75), font, 0.75, (0, 0, 255), 2, cv2.LINE_AA)
```

PREDICT IMAGE

```
predictions = model.predict(img)
```

```
classIndex = model.predict_classes(img)
```

```
probabilityValue = np.amax(predictions)
```

```
if probabilityValue > threshold:
```

```
#print(getCalssName(classIndex))
```

```
cv2.putText(imgOriginal, str(classIndex)+" "+str(getCalssName(classIndex)), (120, 35), font, 0.75, (0, 0, 255), 2, cv2.LINE_AA)
```

```
cv2.putText(imgOriginal, str(round(probabilityValue*100,2))+"%", (180, 75), font, 0.75, (0, 0, 255), 2, cv2.LINE_AA)
```

```
cv2.imshow("Result", imgOriginal)
```

if cv2.waitKey(1) and 0xFF == ord('q'):

break

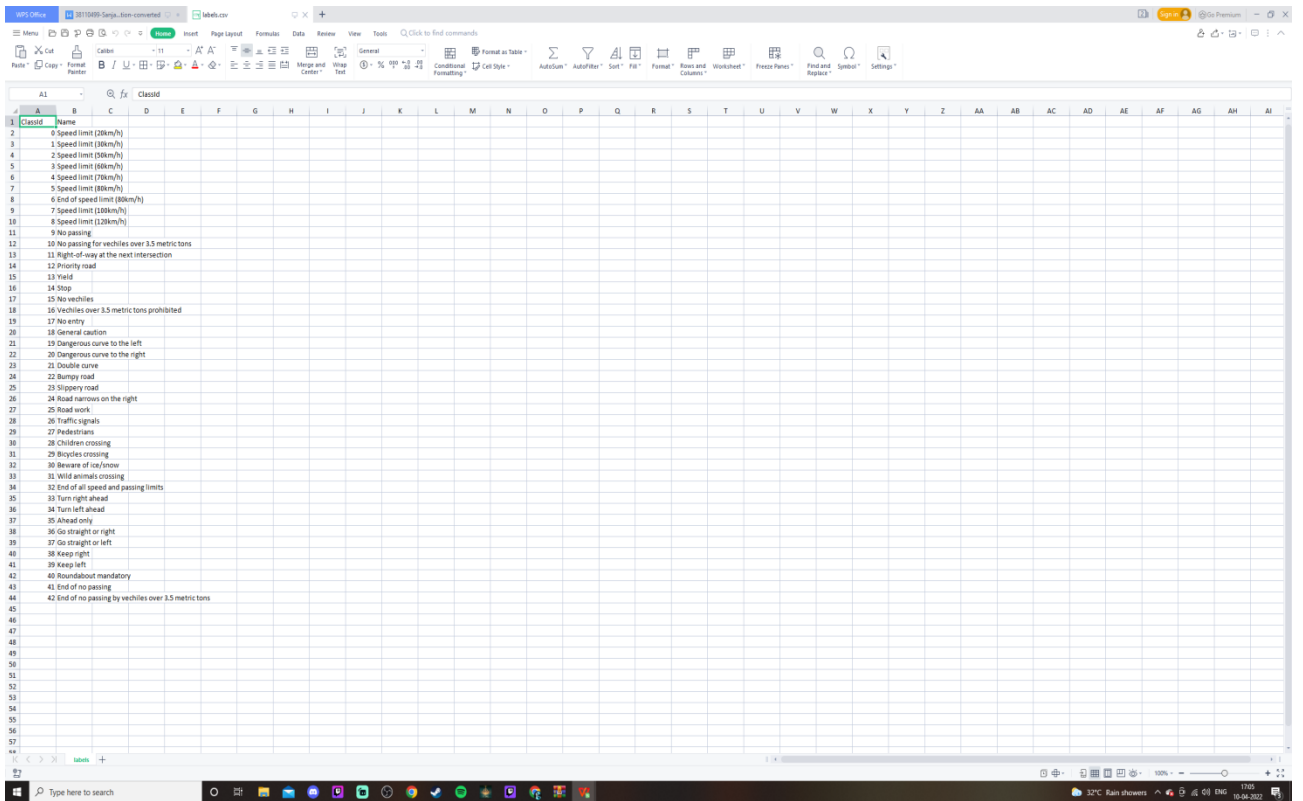
SCREENSHOTS



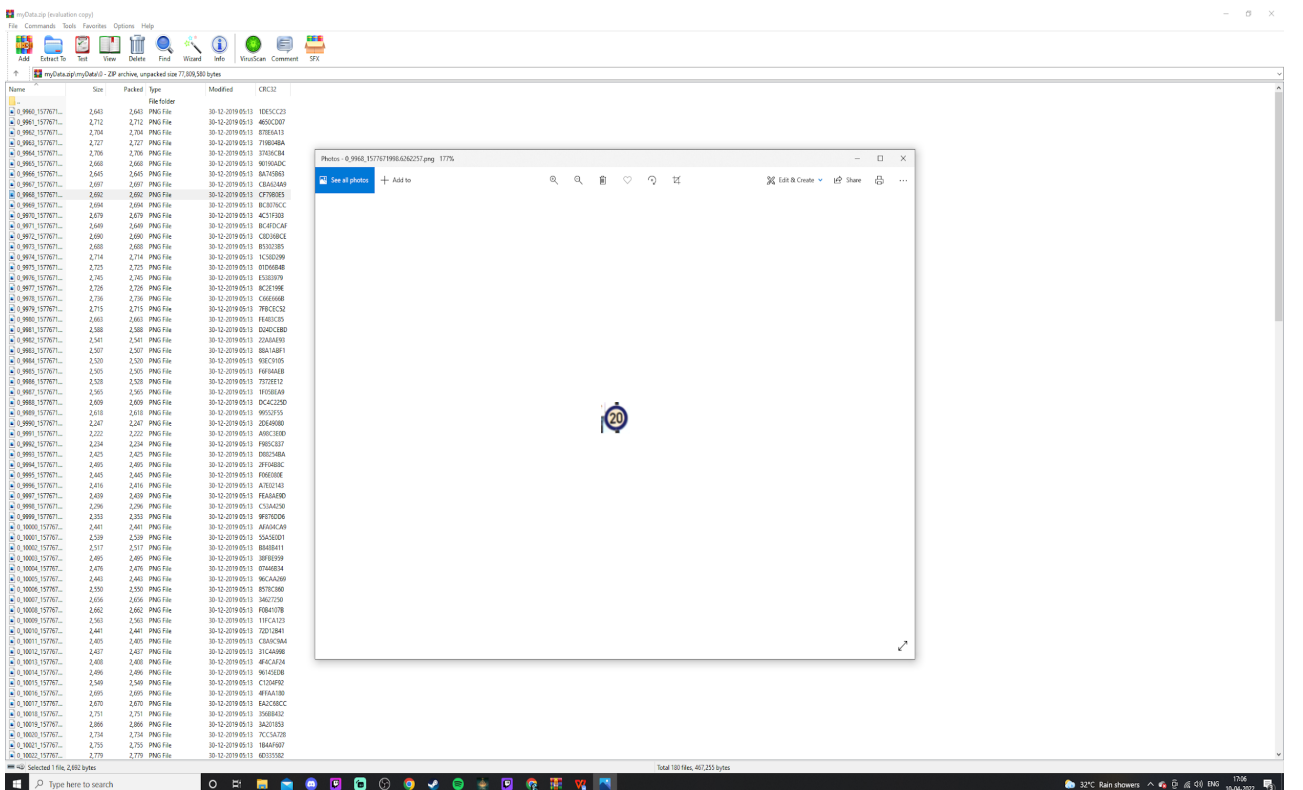
Jupyter Notebook Home Page



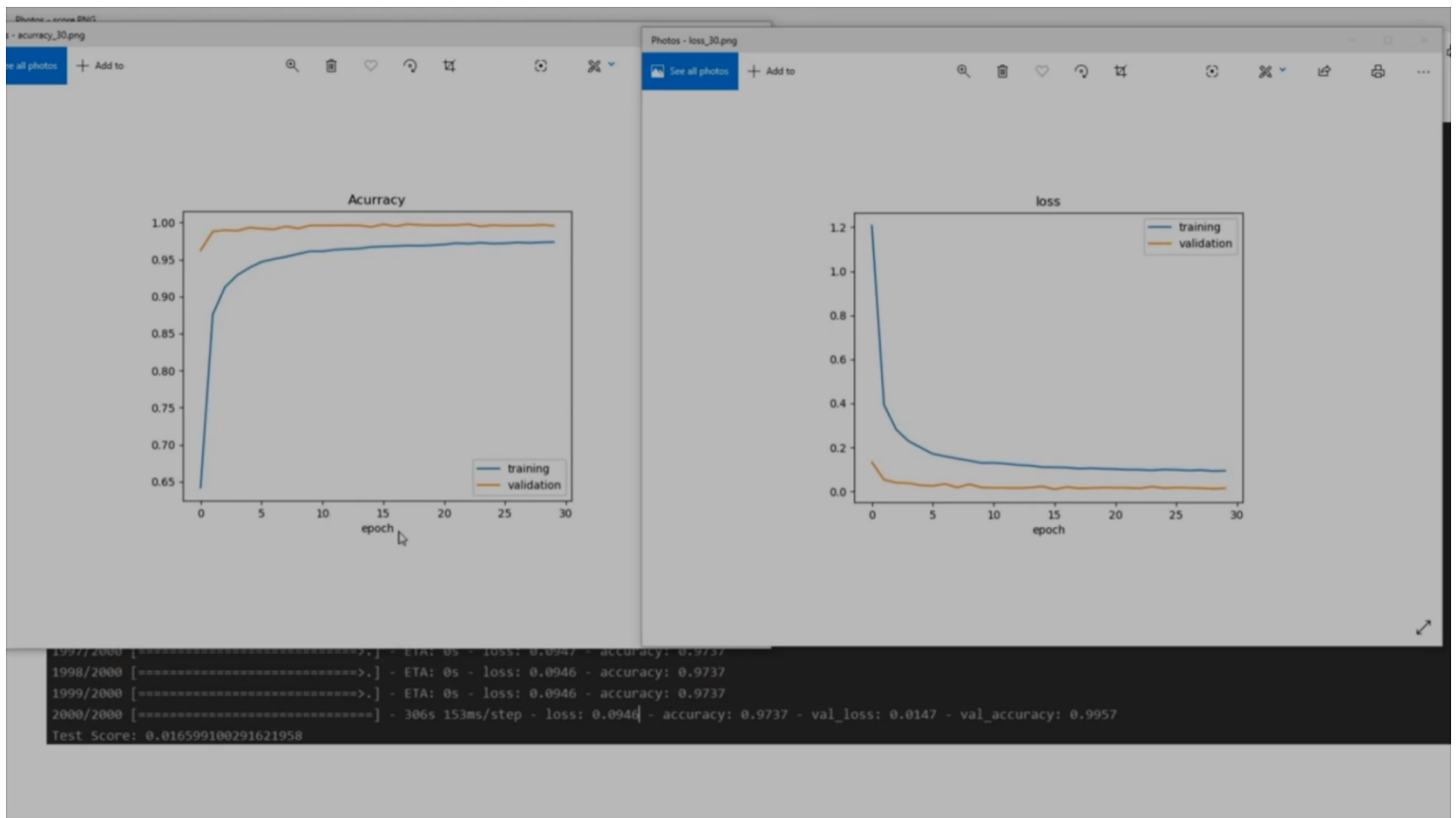
Jupyter Notebook Environment



Labels Preview



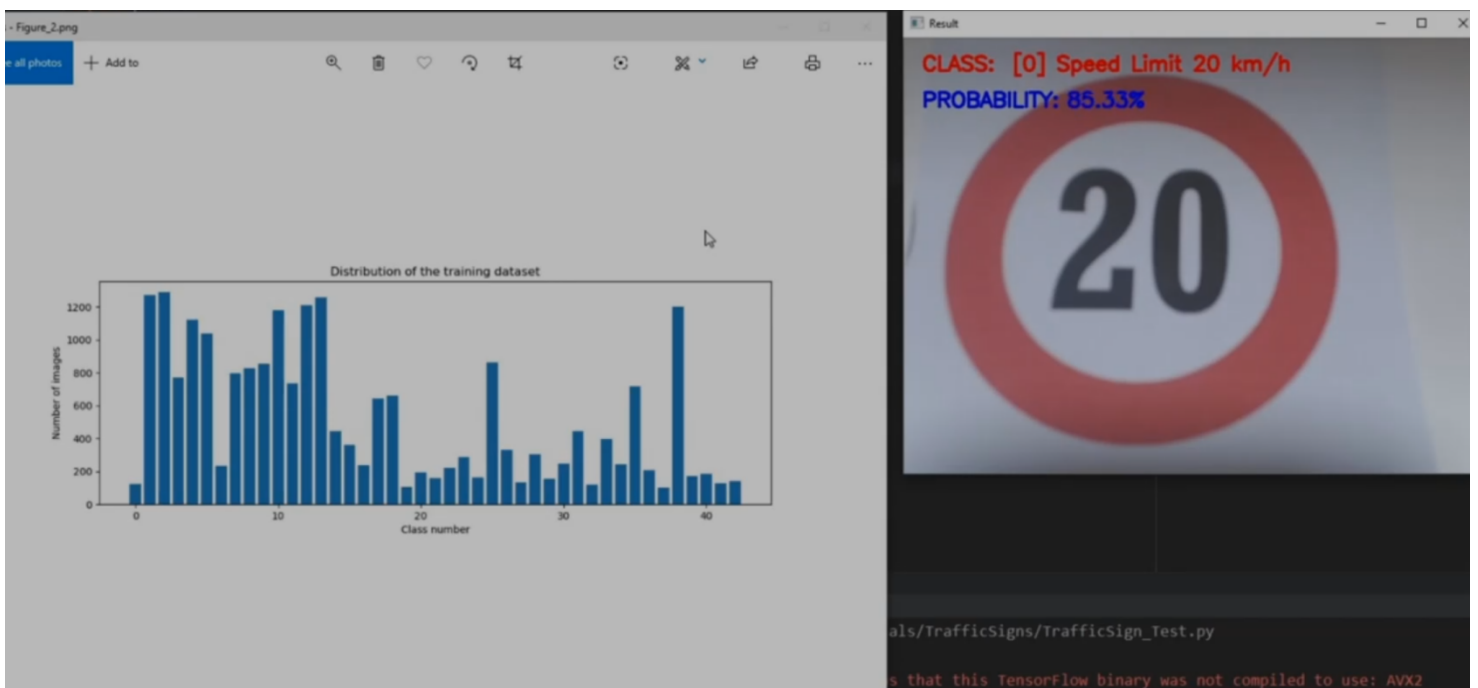
Dataset Preview



Analysis Graph Output



Preprocessed Images



Camera Output