MACHINE LEARNING

(Student Intervention)

Summer Internship Report Submitted in partial fulfillment of the requirement for undergraduate degree of

Bachelor of Technology

In

COMPUTER SCIENCE ENGINEERING

By

KUNA KAVYA

221710315023

Under the Guidance of

Department of Computer Science Engineering
GITAM School of Technology
GITAM (Deemed to be University)
Hyderabad-502329
July 2020

DECLARATION

I submit this industrial training work entitled **“ TRAFFIC SIGN RECOGNITION ”** to GITAM (Deemed To Be University) , Hyderabad in partial fulfillment of the requirements for the award of the degree of **“ Bachelor of Technology ”** in **“ Computer Science Engineering ”**. I declare that it was carried out independently by me under the guidance of **Mr……..**, Asst. Professor, GITAM (Deemed To Be University), Hyderabad, India.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD kuna kavya

Date: 221710315023

GITAM (DEEMED TO BE UNIVERSITY)

Hyderabad-502329, India

Dated:

CERTIFICATE

This is to certify that the Industrial Training Report entitled **" TRAFFIC SIGN RECOGNITION"** is being submitted by kuna kavya (221710315023) in partial fulfillment of the requirement for the award of **Bachelor of Technology in Computer Science Engineering** at GITAM (Deemed To Be University), Hyderabad during the academi year 2020-21.

It is faithful record work carried out by her at the **Computer Science Engineering Department,** GITAM University Hyderabad Campus under my guidance and supervision.

Mr……
Assistant Professor
Department of CSE

Dr.S.Phani Kumar
Professor and HOD
Department of CSE

ACKNOWLEDGEMENT

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful competition of this internship.

I would like to thank respected **Dr. N. Siva Prasad**, Pro Vice Chancellor, GITAM Hyderabad and **Dr. CH. Sanjay**, Principal, GITAM Hyderabad.

I would like to thank respected **Dr. S.Phani Kumar** , Head of the Department of Computer Science Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present an internship report. It helped me a lot to realize what we study for.

I would like to thank the respected faculties **Mr. ……..** who helped me to make this internship a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

Kuna kavya

221710315023

ABSTRACT

Machine learning algorithms are used to predict the values from the data set by splitting the data set in to train and test and building Machine learning algorithms models of higher accuracy to predict the values is the primary task to be performed on the Train and Test data set. My perception of understanding the given data sets has been in the view of undertaking different activities performed by 30 subjects and prediction of the activity.

To get a better understanding and work on a strategic approach for solution of the client, I have adapted to look at the activities performed by plotting few graphs to understand it in a better way, for further deep understanding of the problem, I have taken the describe function to know more about my data set , and my primary objective of this case study was to predict the activity performed with at most accuracy, so I have used several algorithms to know which one yields the highest accuracy to predict my outcome.

Table of Contents

CHAPTER 1
MACHINE LEARNING

1.1 INTRODUCTION:

Machine Learning(ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of Artificial Intelligence(AI).

1.2 IMPORTANCE OF MACHINE LEARNING:

Consider some of the instances where machine learning is applied: the self-driving Google car, cyber fraud detection, online recommendation engines—like friend suggestions on Facebook, Netflix showcasing the movies and shows you might like, and "more items to consider" and "get yourself a little something" on Amazon—are all examples of applied machine learning. All these examples echo the vital role machine learning has begun to take in today's data-rich world.

Machines can aid in filtering useful pieces of information that help in major advancements, and we are already seeing how this technology is being implemented in a wide variety of industries.

With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the last few years; that's in part due to increased sophistication of machine learning, which helps analyze those big chunks of big data.

Machine learning has also changed the way data extraction, and interpretation is done by involving automatic sets of generic methods that have replaced traditional statistical techniques.

The process flow depicted here represents how machine learning works.

Figure 1 : The Process Flow

1.3 USES OF MACHINE LEARNING:

Earlier in this article, we mentioned some applications of machine learning. To understand the concept of machine learning better, let's consider some more examples: web search results, real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition. All these are by-products of applying machine learning to analyze huge volumes of data.

Traditionally, data analysis was always being characterized by trial and error, an approach that becomes impossible when data sets are large and heterogeneous. Machine learning comes as the solution to all this chaos by proposing clever alternatives to analyzing huge volumes of data.

By developing fast and efficient algorithms and data-driven models for real-time processing of data, machine learning can produce accurate results and analysis

1.4 TYPES OF LEARNING ALGORITHMS:

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

1.4.1 Supervised Learning:

When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category of supervised learning.Supervised machine learning algorithms uncover insights, patterns, and relationships from a labelled training dataset – that is, a dataset that already contains a known value for the target variable for each record. Because you provide the machine learning algorithm with the correct answers for a problem during training, it is able to "learn" how the rest of the features relate to the target, enabling you to uncover insights and make predictions about future outcomes based on historical data.

Examples of Supervised Machine Learning Techniques are Regression, in which the algorithm returns a numerical target for each example, such as how much revenue will be generated from a new marketing campaign.

Classification, in which the algorithm attempts to label each example by choosing between two or more different classes. Choosing between two classes is called binary classification, such as determining whether or not someone will default on a loan. Choosing between more than two classes is referred to as multiclass classification.

1.4.2 Unsupervised Learning:

When an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own.

This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of uncorrelated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.

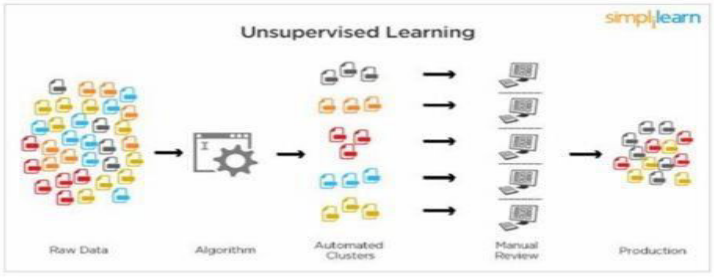


Figure 2 : Unsupervised Learning

Popular techniques where unsupervised learning is used also include self - organizing maps, nearest neighbor mapping, singular value decomposition, and k-means clustering. Basically, online recommendations, identification of data outliers, and segment text topics are all examples of unsupervised learning.

1.4.3 Semi Supervised Learning:

As the name suggests, semi-supervised learning is a bit of both supervised and unsupervised learning and uses both labeled and unlabeled data for training. In a typical scenario, the algorithm would use a small amount of labeled data with a large amount of unlabeled data.

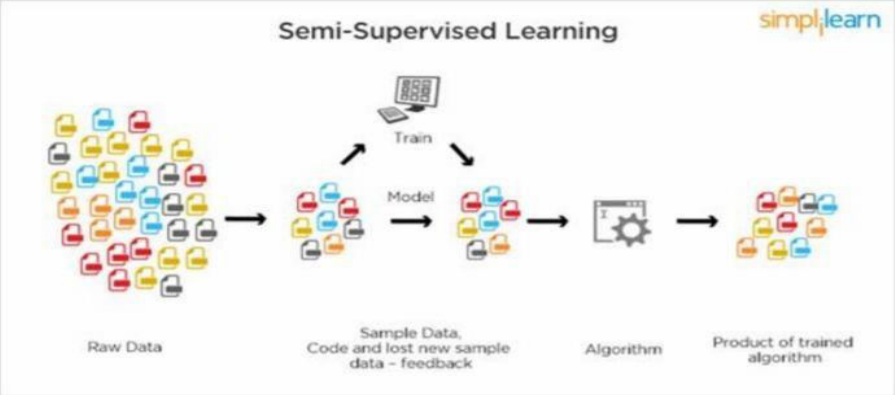


Figure 3 : Semi Supervised Learning

1.5 RELATION BETWEEN DATA MINING,MACHINE LEARNING AND DEEP LEARNING:

Machine learning and data mining use the same algorithms and techniques as data mining, except the kinds of predictions vary. While data mining discovers previously unknown patterns and knowledge, machine learning reproduces known patterns and knowledge—and further automatically applies that information to data, decision-making, and actions.

Deep learning, on the other hand, uses advanced computing power and special 5 types of neural networks and applies them to large amounts of data to learn, understand, and identify complicated patterns. Automatic language translation and medical diagnoses are examples of deep learning.

CHAPTER 2

PYTHON

Basic programming language used for machine learning is : PYTHON

2.1 INTRODUCTION TO PYHTON:

- Python is a high-level, interpreted, interactive and object-oriented scripting language.
- Python is a general purpose programming language that is often applied in scripting roles.
- Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.
- Python is Interactive: You can sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented: Python supports the Object-Oriented style or technique of programming that encapsulates code within objects.

2.2 HISTORY OF PYTHON:

- Python was developed by GUIDO VAN ROSSUM in early 1990's.
- Its latest version is 3.7 , it is generally called as python3

2.3 FEATURES OF PYTHON:

- Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax, This allows the student to pick up the language quickly.
- Easy-to-read: Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain: Python's source code is fairly easy-to-maintaining.
- A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

- Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- Databases: Python provides interfaces to all major commercial databases.
- GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

2.4 HOW TO SETUP PYTHON:

- Python is available on a wide variety of platforms including Linux and Mac OS X.

  Let's understand how to set up our Python environment.

- The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python.

2.4.1 Installation(using python IDLE):

- Installing python is generally easy, and nowadays many Linux and Mac OS distributions include a recent python.
- Download python from www.python.org
- When the download is completed, double click the file and follow the instructions to install it.
- When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.

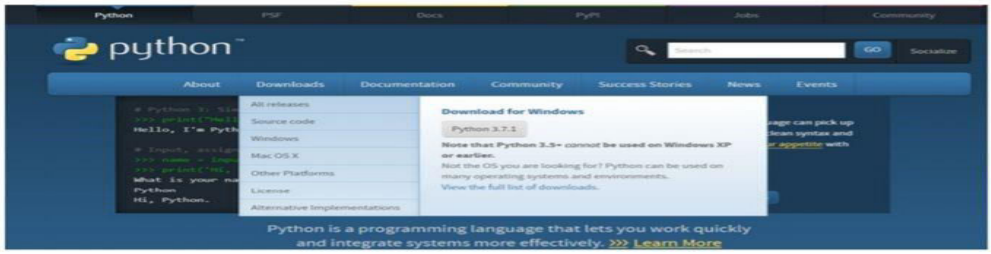


Figure 4 . Python download

2.4.2 Installation(using Anaconda):

- Python programs are also executed using Anaconda.
- Anaconda is a free open source distribution of python for large scale data processing, predictive analytics and scientific computing.
- Annconda is a package manager quickly installs and manages packages.
- In WINDOWS:
- In window:
- Step 1: Open Anaconda.com/downloads in web browser.
- Step 2: Download python 3.4 version for (32-bitgraphic installer/64 -bit graphic installer)
- Step 3: select installation type(all users)
- Step 4: Select path(i.e. add anaconda to path & register anaconda default python 3.4) next click install and next click finish.
- Step 5: Open jupyter notebook (it opens in default browser)

Figure 5 : Anaconda download

Figure 6 : Jupyter notebook

2.5 PYTHON VARIABLE TYPES:

- Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.
- Variables are nothing but reserved memory locations to store values.
- Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory.
- Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.
- Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.
- Python has five standard data types –
  Number, Strings , Lists ,Tuples, Dictionary

2.5.1 Python Numbers:

- Number data types store numeric values. Number objects are created when you assign a value to them.
- Python supports four different numerical types − int (signed integers) long (long integers, they can also be represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

2.5.2 Python Strings:

- Strings in Python are identified as a contiguous set of characters represented in the quotation marks
- Python allows for either pairs of single or double quotes.
- Subsets of strings can be taken using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.
- The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

2.5.3 Python Lists:

- Lists are the most versatile of Python's compound data types.
- A list contains items separated by commas and enclosed within square brackets .
- To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.
- The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.

- The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

2.5.4 Python Tuples:

- A tuple is another sequence data type that is similar to the list.
- A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.
- The main differences between lists and tuples are: Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated.
- Tuples can be thought of as read-only lists.
- For example − Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't add elements to a tuple. Tuples have no append or extend method. You can't remove elements from a tuple. Tuples have no remove or pop method.

2.5.5 Python Dictionary:

- Python's dictionaries are kind of hash table type. They work like associative arrays 12 or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.
- Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).
- You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.
- What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing to another, no matter what it is.

2.6 PYTHON FUNCTION:

2.6.1 Defining a Function:

You can define functions to provide the required functionality. Here are simple rules to define a function in Python. Function blocks begin with the keyword def followed by the function name and parentheses (i.e.()).

Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.

The code block within every function starts with a colon (:) and is indented. The statement returns [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None

2.6.2 Calling a Function:

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.

2.7PYTHON USING OOP's CONCEPTS:

2.7.1 Class:

- Class: A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.
- Class variable: A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.
- Data member: A class variable or instance variable that holds data associated with a class and its objects.
- Instance variable: A variable that is defined inside a method and belongs only to the current instance of a class.
- Defining a Class:We define a class in a very similar way how we define a function. Just like a function ,we use parentheses and a colon after the class name(i.e. ():) when we define a class. Similarly, the body of our class is 14 indented like a functionns body is.

```
def my_function():
    # the details of the
    # function go here
```

```
class MyClass():
    # the details of the
    # class go here
```

Figure 7 : Defining a Class

2.7.2__init__ method in Class:

- The init method — also called a constructor — is a special method that runs when an instance is created so we can perform any tasks to set up the instance.
- The init method has a special name that starts and ends with two underscores:__init__().

CHAPTER 3
CASE STUDY

3.1 PROBLEM STATEMENT:

TRAFFIC SIGN RECOGNITION

Self-driving cars need traffic sign recognition in order to properly parse and understand the roadway. Similarly, “driver alert” systems inside cars need to understand the roadway around them to help aid and protect drivers.

Traffic sign recognition is just one of the problems that computer vision and deep learning can solve.

3.2 DATA SET:

we have my data that will be available in a different link so if we look in our data we can see we have 43 different folders starting form 0 and ending to 42. So in each folder we have images of the relevant classes. In the labels we have the names of these classes so wt we have seen back is the IDs 0 1 2 till 42 but in our labels we have the name of each I'd so 0 represents speed limit of 20 whereas 14 represents stop and fourth…

3.3 OBJECTIVE OF THE CASE STUDY:

Traffic-sign recognition (TSR) is a technology by which a vehicle is able to recognize the traffic signs put on the road e.g. "speed limit" or "children" or "turn ahead". This is part of the features collectively called ADAS. The technology is being developed by a variety of automotive suppliers. It uses image processing techniques to detect the traffic signs. The detection methods can be generally divided into color based, shape based and learning based methods.

CHAPTER 4
PROJECT

4.1 GETTING THE DATA:
4.1.1 CLONING BY GIT:

```
[ ]  !git clone https://bitbucket.org/jadslim/german-traffic-signs

     Cloning into 'german-traffic-signs'...
     remote: Counting objects: 6, done.
     remote: Compressing objects: 100% (6/6), done.
     remote: Total 6 (delta 0), reused 0 (delta 0)
     Unpacking objects: 100% (6/6), done.
```

FIGURE 4.1.1.1 : DATASET LINK

4.1.2 LISTIG OUT FIES:

```
[ ]  !ls german-traffic-signs

     signnames.csv  test.p  train.p  valid.p
```

FIGURE 4.1.2.1 : LIST OF FILES

4.1.3 IMPORTING THE LIBRARIES:

```
[ ]  import numpy as np
     import matplotlib.pyplot as plt
     import keras
     from keras.models import Sequential
     from keras.optimizers import Adam
     from keras.layers import Dense
     from keras.layers import Flatten, Dropout
     from keras.utils.np_utils import to_categorical
     from keras.layers.convolutional import Conv2D, MaxPooling2D
     import random
     import pickle
     import pandas as pd
     import cv2
```

```
[ ]  np.random.seed(0)
```

FIGURE 4.1.3.1 : LIBRARIES

- **numpy** package can be used to perform mathematical operations like 'mean'.
- **Matplotlib:** Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.
- **Keras** is an open source neural network library written in Python. It is capable of running on top of TensorFlow. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.
- **Sequential model** is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.
- **Conv2D is a** layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. If `use_bias` is True, a bias vector is created and added to the outputs. Finally, if `activation` is not `None`, it is applied to the outputs as well.
- **MaxPooling2D:** Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.)
- **to_categorical** that Converts a class vector (integers) to binary class matrix.
- **Dense** implements the operation: `output = activation(dot(input, kernel) + bias)` where `activation` is the element-wise activation function passed as the `activation` argument, `kernel` is a weights matrix created by the layer, and `bias` is a bias vector created by the layer (only applicable if `use_bias` is `True`).
- **pandas** package can be used to process dataframes.
- **flatten** operation on a tensor reshapes the tensor to have the shape that is equal to the number of elements contained in tensor non including the batch dimension.

  Here, we are using numpy for numerical computations, pandas for importing and managing the dataset, Keras for building the Convolutional Neural Network quickly with less code, cv2 for doing some preprocessing steps which are necessary for efficient extraction of features from the images by the CNN.

4.1.4 LOADING THE DATA AS TRAIN, TEST, VALIDATION:

Time to load the data. We will use pandas to load signnames.csv, and pickle to load the train, validation and test pickle files. After extraction of data, it is then split using the dictionary labels “features” and “labels”.

```
[ ]  with open('german-traffic-signs/train.p', 'rb') as f:
         train_data = pickle.load(f)
     with open('german-traffic-signs/valid.p', 'rb') as f:
         val_data = pickle.load(f)
     # TODO: Load test data
     with open('german-traffic-signs/test.p', 'rb') as f:
         test_data = pickle.load(f)

     X_train, y_train = train_data['features'], train_data['labels']
     X_val, y_val = val_data['features'], val_data['labels']
     X_test, y_test = test_data['features'], test_data['labels']

     print(X_train.shape)
     print(X_test.shape)
     print(X_val.shape)
```

```
(34799, 32, 32, 3)
(12630, 32, 32, 3)
(4410, 32, 32, 3)
```

FIGURE 4.1.4.1 : LOADING DATA

```
[ ]  assert(X_train.shape[0] == y_train.shape[0]), "The number of images is not equal to the number of labels."
     assert(X_train.shape[1:] == (32,32,3)), "The dimensions of the images are not 32 x 32 x 3."
     assert(X_val.shape[0] == y_val.shape[0]), "The number of images is not equal to the number of labels."
     assert(X_val.shape[1:] == (32,32,3)), "The dimensions of the images are not 32 x 32 x 3."
     assert(X_test.shape[0] == y_test.shape[0]), "The number of images is not equal to the number of labels."
     assert(X_test.shape[1:] == (32,32,3)), "The dimensions of the images are not 32 x 32 x 3."
```

4.1.5 VISUALIZING THE DATA:

There are several different types of traffic signs like speed limits, no entry, traffic signals, turn left or right, children crossing, no passing of heavy vehicles, etc. Traffic signs classification is the process of identifying which class a traffic sign belongs to.

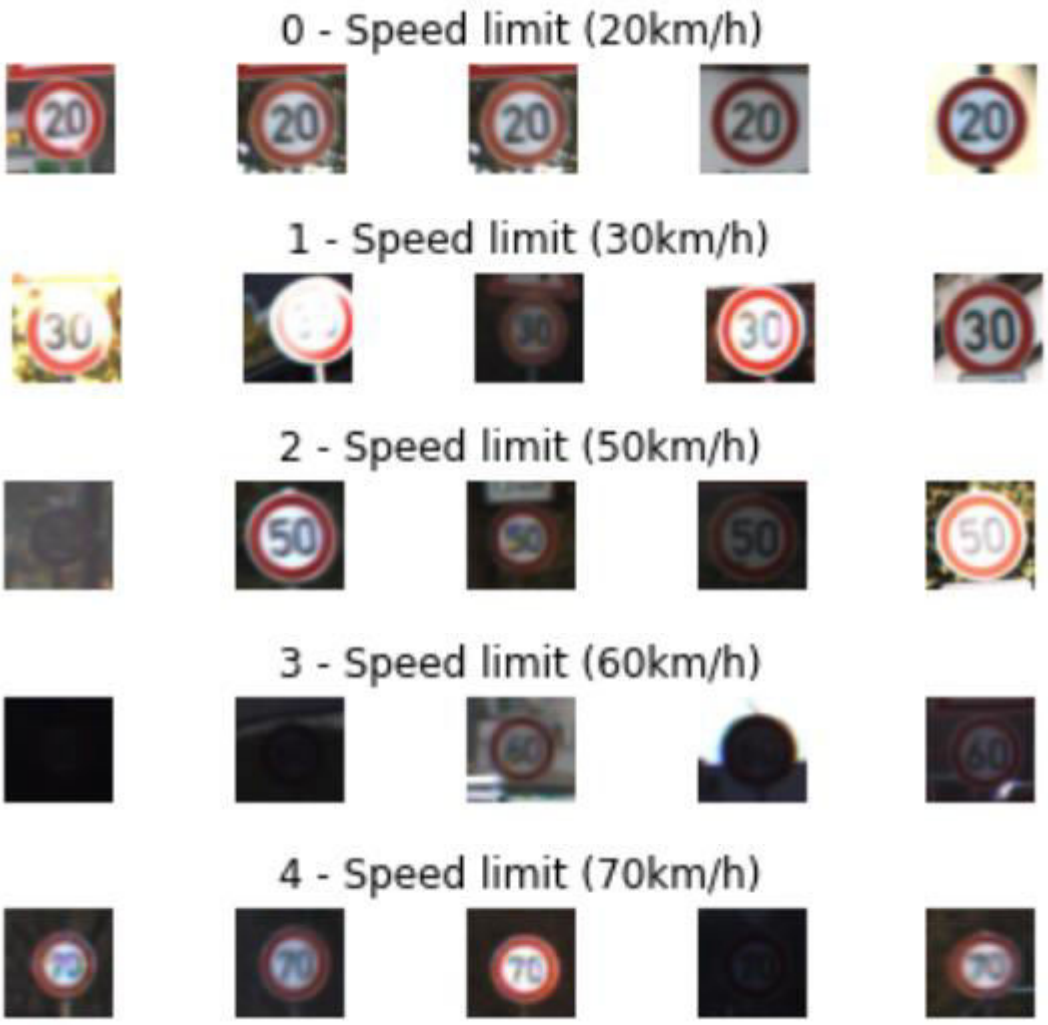
we will build a deep neural network model that can classify traffic signs present in the image into different categories. With this model, we are able to read and understand traffic signs which are a very important task for all autonomous vehicles.

```
[ ]  data = pd.read_csv('german-traffic-signs/signnames.csv')
     num_of_samples=[]

     cols = 5
     num_classes = 43

     fig, axs = plt.subplots(nrows=num_classes, ncols=cols, figsize=(5,50))
     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["SignName"])
             num_of_samples.append(len(x_selected))
```



5 - Speed limit (80km/h)

6 - End of speed limit (80km/h)

7 - Speed limit (100km/h)

8 - Speed limit (120km/h)

9 - No passing

10 - No passing for vechiles over 3.5 metric tons

11 - Right-of-way at the next intersection

12 - Priority road

13 - Yield

14 - Stop

15 - No vechiles

16 - Vechiles over 3.5 metric tons prohibited

17 - No entry

18 - General caution

19 - Dangerous curve to the left

20 - Dangerous curve to the right

21 - Double curve

22 - Bumpy road

23 - Slippery road

24 - Road narrows on the right

25 - Road work

26 - Traffic signals

27 - Pedestrians

28 - Children crossing

29 - Bicycles crossing

30 - Beware of ice/snow

31 - Wild animals crossing

32 - End of all speed and passing limits

33 - Turn right ahead

34 - Turn left ahead

35 - Ahead only

36 - Go straight or right

FIGURE 4.1.5.1 : VISUALIZING

```
[ ]  print(num_of_samples)
     plt.figure(figsize=(12, 4))
     plt.bar(range(0, num_classes), num_of_samples)
     plt.title("Distribution of the train dataset")
     plt.xlabel("Class number")
     plt.ylabel("Number of images")
     plt.show()
```

```
[180, 1980, 2010, 1260, 1770, 1650, 360, 1290, 1260, 1320, 1800, 1170, 1890, 1920, 690, 540, 360, 990, 1080,
```

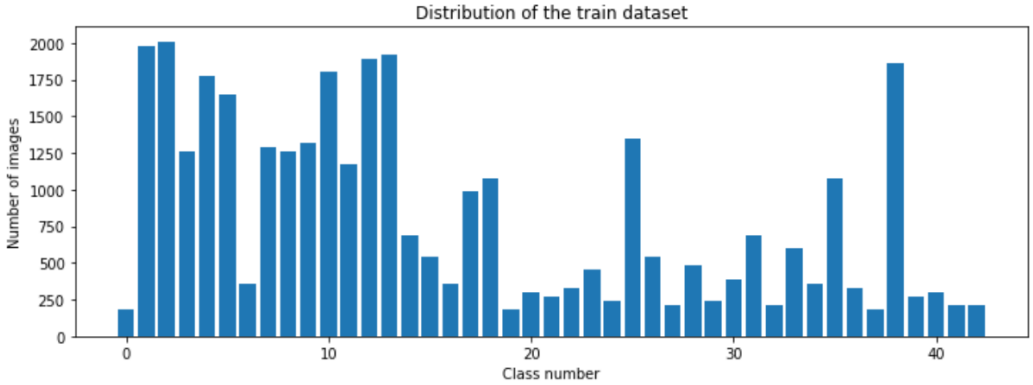


FIGURE 4.1.5.2: DISTRIBUTION O TRAIN DATA SET

4.1.6 USING OPENCV2 FOR PREPROCESSING THE IMAGE:

```
[ ]  import cv2

     plt.imshow(X_train[1000])
     plt.axis("off")
     print(X_train[1000].shape)
     print(y_train[1000])
     def grayscale(img):
         img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
         return img
```

```
(32, 32, 3)
36
```

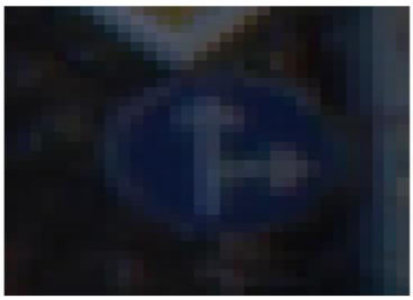
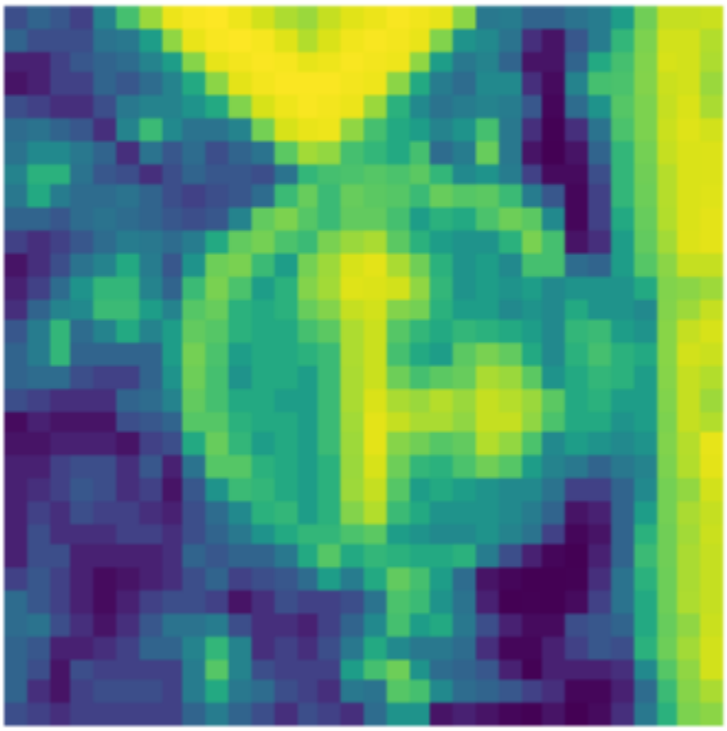


FIGURE 4.1.6.1 : PREPROCESSING IMAGE

```
[ ]  img = grayscale(X_train[1000])
     plt.imshow(img)
     plt.axis("off")
     print(img.shape)
     def equalize(img):
         img = cv2.equalizeHist(img)
         return img
     img = equalize(img)
     plt.imshow(img)
     plt.axis("off")
     print(img.shape)
```

```
(32, 32)
(32, 32)
```



Preprocessing the data using OpenCV

Preprocessing images before feeding into the model gives very accurate results as it helps in extracting the complex features of the image. OpenCV has some built-in functions like **g**rayscale() and equalize() for this task. Follow the below steps for this task –

- First, the images are converted to grayscale images for reducing computation using the grayscale() function.

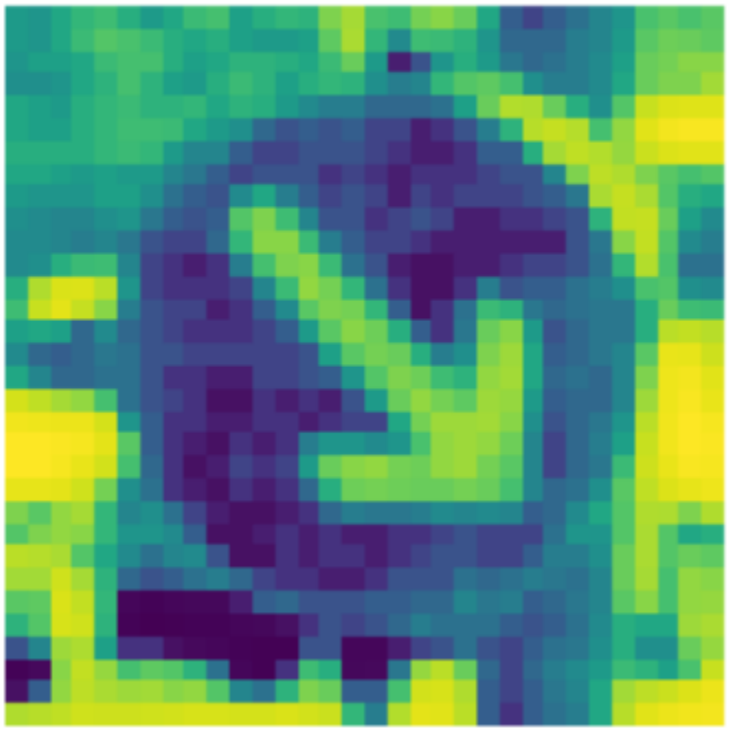
- The equalize() function increases the contrasts of the image by equalizing the intensities of the pixels by normalizing them with their nearby pixels.
- At the end, we normalize the pixel values between 0 and 1 by dividing them by 255.

```
[ ]  def preprocess(img):
         img = grayscale(img)
         img = equalize(img)
         img = img/255
         return img
```

```
[ ]  X_train = np.array(list(map(preprocess, X_train)))
     X_test = np.array(list(map(preprocess, X_test)))
     X_val = np.array(list(map(preprocess, X_val)))

     plt.imshow(X_train[random.randint(0, len(X_train) - 1)])
     plt.axis('off')
     print(X_train.shape)
```

```
(34799, 32, 32)
```



```
[ ]  X_train = X_train.reshape(34799, 32, 32, 1)
     X_test = X_test.reshape(12630, 32, 32, 1)
     X_val = X_val.reshape(4410, 32, 32, 1)
```

After reshaping the arrays, it's time to feed them into the model for training. But to increase the accuracy of our CNN model, we will involve one more step of generating augmented images using the ImageDataGenerator.

This is done to reduce overfitting the training data as getting more varied data will result in a better model. The value 0.1 is interpreted as 10%, whereas 10 is the degree of rotation. We are also converting the labels to categorical values, as we normally do.

```
[ ]  from keras.preprocessing.image import ImageDataGenerator

     datagen = ImageDataGenerator(width_shift_range=0.1,
                                  height_shift_range=0.1,
                                  zoom_range=0.2,
                                  shear_range=0.1,
                                  rotation_range=10.)

     datagen.fit(X_train)
     batches = datagen.flow(X_train, y_train, batch_size = 15)
     X_batch, y_batch = next(batches)

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

     for i in range(15):
         axs[i].imshow(X_batch[i].reshape(32, 32))
         axs[i].axis("off")

     print(X_batch.shape)
```

```
(15, 32, 32, 1)
```

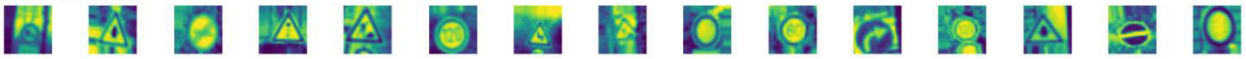


FIGURE 4.1.6.2 : USING KERAS

```
[ ]  y_train = to_categorical(y_train, 43)
     y_test = to_categorical(y_test, 43)
     y_val = to_categorical(y_val, 43)
```

4.2 DEFINING MODEL:

4.2.1 CNN MODEL:

we have 43 classes of images in the dataset. The model contains two Conv2D layers followed by one MaxPooling2D layer. This is done two times for the effective extraction of features, which is followed by the Dense layers. A dropout layer of 0.5 is added to avoid overfitting the data.

```
[ ]  def modified_model():
       model = Sequential()
       model.add(Conv2D(60, (5, 5), input_shape=(32, 32, 1), activation='relu'))
       model.add(Conv2D(60, (5, 5), activation='relu'))
       model.add(MaxPooling2D(pool_size=(2, 2)))

       model.add(Conv2D(30, (3, 3), activation='relu'))
       model.add(Conv2D(30, (3, 3), activation='relu'))
       model.add(MaxPooling2D(pool_size=(2, 2)))

       model.add(Flatten())
       model.add(Dense(500, activation='relu'))
       model.add(Dropout(0.5))
       model.add(Dense(43, activation='softmax'))

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

FIGURE 4.2.1.1 : USING CNN MODEL

4.2.2 FITING THE MODEL

```
[ ]  model = modified_model()
     print(model.summary())

     history = model.fit_generator(datagen.flow(X_train, y_train, batch_size=50),
                                   epochs=10,
                                   validation_data=(X_val, y_val), shuffle = 1)
```

FIGURE 4.2.2.1 : FITING

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 60)        1560
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 60)        90060
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 60)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 30)        16230
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 30)          8130
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 30)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 480)               0
_________________________________________________________________
dense_1 (Dense)              (None, 500)               240500
_________________________________________________________________
dropout_1 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_2 (Dense)              (None, 43)                21543
=================================================================
Total params: 378,023
Trainable params: 378,023
Non-trainable params: 0
```

```
_________________________________________________________________
None
Epoch 1/10
696/696 [==============================] - 362s 520ms/step - loss: 1.7561 - accuracy: 0.5020 - val_loss: 0.3348 - val_accuracy: 0.8900
Epoch 2/10
696/696 [==============================] - 361s 519ms/step - loss: 0.5026 - accuracy: 0.8451 - val_loss: 0.1390 - val_accuracy: 0.9608
Epoch 3/10
696/696 [==============================] - 360s 517ms/step - loss: 0.2970 - accuracy: 0.9066 - val_loss: 0.0898 - val_accuracy: 0.9707
Epoch 4/10
696/696 [==============================] - 361s 519ms/step - loss: 0.2266 - accuracy: 0.9300 - val_loss: 0.0777 - val_accuracy: 0.9814
Epoch 5/10
696/696 [==============================] - 359s 515ms/step - loss: 0.1825 - accuracy: 0.9436 - val_loss: 0.0533 - val_accuracy: 0.9850
Epoch 6/10
696/696 [==============================] - 360s 517ms/step - loss: 0.1589 - accuracy: 0.9514 - val_loss: 0.0509 - val_accuracy: 0.9864
Epoch 7/10
696/696 [==============================] - 360s 518ms/step - loss: 0.1304 - accuracy: 0.9601 - val_loss: 0.0349 - val_accuracy: 0.9900
Epoch 8/10
696/696 [==============================] - 362s 521ms/step - loss: 0.1280 - accuracy: 0.9599 - val_loss: 0.0416 - val_accuracy: 0.9875
Epoch 9/10
696/696 [==============================] - 362s 520ms/step - loss: 0.1127 - accuracy: 0.9653 - val_loss: 0.0472 - val_accuracy: 0.9841
Epoch 10/10
696/696 [==============================] - 361s 519ms/step - loss: 0.1041 - accuracy: 0.9675 - val_loss: 0.0401 - val_accuracy: 0.9880
```

After successfully compiling the model, and fitting in on the train and validation data, let us evaluate it by using Matplotlib.We've been able to reach a maximum accuracy of **98.8%** on the validation set over 10 epochs.

4.3 PLOT THE ACCURACY GRAPH:

With the help of matplotlib functions, we will plot the graph of training and validation accuracy.

4.3.1 DISPLAYING THE LOSS VALUES:

```
[36] import matplotlib.pyplot as plt
     plt.figure(0)
     plt.plot(history.history['loss'], label='training loss')
     plt.plot(history.history['val_loss'], label='val loss')
     plt.title('Loss')
     plt.xlabel('epochs')
     plt.ylabel('loss')
     plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f2eac8e7eb8>
```

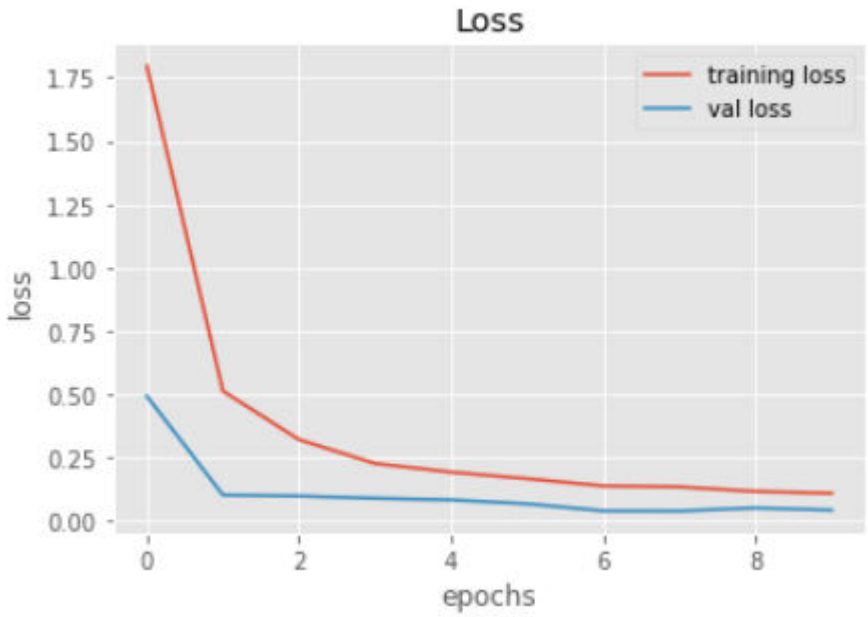


FIGURE 4.3.1.1 :LOSS VALES

4.3.2 DISPLAYING THE ACCURACY VALUES

```
plt.figure(1)
plt.plot(history.history['accuracy'], label='training accuracy')
plt.plot(history.history['val_accuracy'], label='val accuracy')
plt.legend(['training','test'])
plt.title('Accuracy')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend()
plt.style.use('ggplot')
```

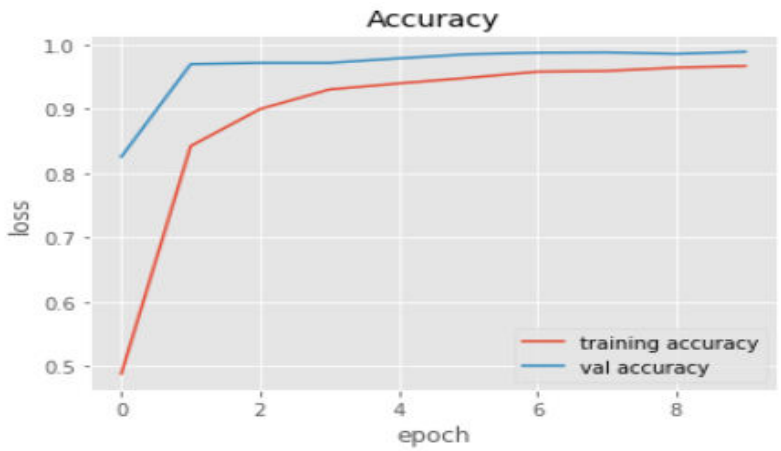


FIGURE 4.3.2.1: ACCURACY VALUE

4.4 TAKING A RANDOM IMAGE FROM INTERNET TO TEST MY MODEL:

4.4.1 DISLAYING RAW IMAGE:

```
[ ] import requests
    from PIL import Image
    url = 'https://thumbs.dreamstime.com/t/road-signs-main-road-sign-blue-background-road-signs-main-road-sign-blue-background-109436823.jpg
    r = requests.get(url, stream=True)
    img = Image.open(r.raw)
    plt.imshow(img, cmap=plt.get_cmap('gray'))
```

```
<matplotlib.image.AxesImage at 0x7f2e22b6d630>
```

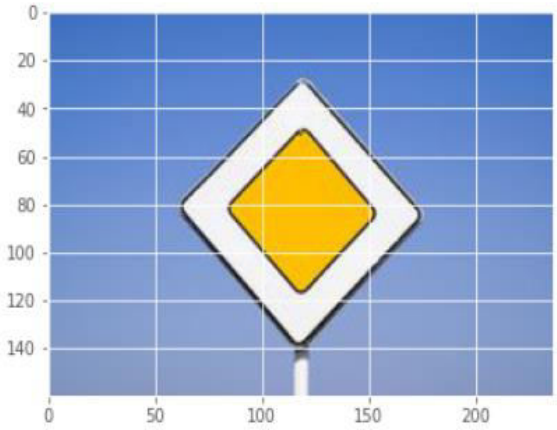


FIGURE 4.4.1.1 : RANDOM IMAGE

4.4.2 DISPLAYING USING CMAP

```
[ ] img = np.asarray(img)
    img = cv2.resize(img, (32, 32))
    img = preprocess(img)
    plt.imshow(img, cmap = plt.get_cmap('gray'))
    print(img.shape)
    img = img.reshape(1, 32, 32, 1)
```

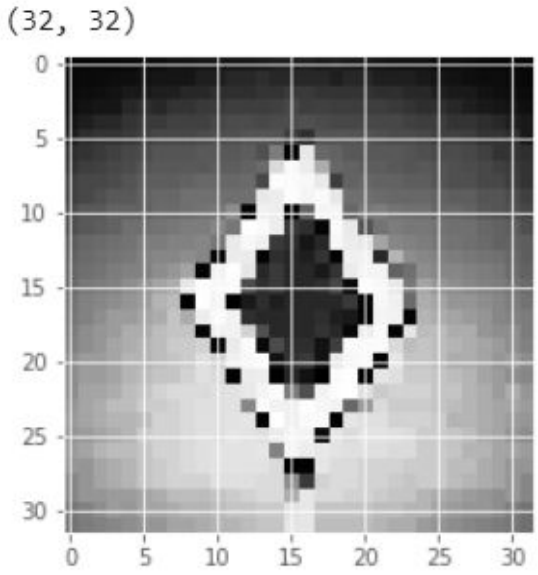


FIGURE 4.4.2.1 : RANDOM IMAGE USING CMAP

CONCLUSION:

I used the CNN model architecture for classifing road symbols. The model seems to work very well as I achieved validation accuracy of around 99% which is really amazing. Also I tried certain random images from the Internet to classify, the model was accuretly able to classify them. I added only a single dropout layer which produced a very good accuracy.

REFERENCES:

https://github.com/kunakavya/traffic-sign-recognition---project

https://colab.research.google.com/drive/10bMK-AWQ-cF2NNDKLnFNWW2psfLmPeN_

https://www.kaggle.com/c/traffic-sign-recognition