ARTIFICIAL INTELLIGENCE (CSE 3013)

FALL SEMESTER (2020-21)

J COMPONENT REPORT

TRAFFIC SIGN AND SIGNAL DETECTION USING CONVOLUTIONAL NEURAL NETWORK

TEAM MEMBERS:
ADITYA ROHILLA (18BCE0929)
KHWAAB THAREJA (18BCE0930)
RAHUL ANAND (18BCE0953)

SUBMITTED TO:
PROF. TAPAN KUMAR DAS

ABSTRACT

Traffic sign and signal recognition (TSSR) represents an important feature of advanced driver assistance systems, contributing to the safety of the drivers, pedestrians and vehicles as well. Developing TSSR systems requires the use of computer vision techniques, which could be considered fundamental in the field of pattern recognition in general. Despite all the previous works and research that has been achieved, traffic sign detection and recognition still remain a very challenging problem, precisely if we want to provide a real time processing solution. We propose an approach for traffic sign and light detection based on Convolutional Neural Networks (CNN). We first transform the original image into the gray scale image by using support vector machines, then use convolutional neural networks with fixed and learnable layers for detection and recognition. The fixed layer can reduce the amount of interest areas to detect, and crop the boundaries very close to the borders of traffic signs. The learnable layers can increase the accuracy of detection significantly.

INDEX

1.Introduction

We propose an approach for traffic sign and signal detection based on Convolutional Neural Networks (CNN). We first transform the original image into the gray scale image by using support vector machines, then use convolutional neural networks with fixed and learnable layers for detection and recognition. The fixed layer can reduce the amount of interest areas to detect, and crop the boundaries very close to the borders of traffic signs. The learnable layers can increase the accuracy of detection significantly. The objective of an automatic road sign and traffic light signal recognition system is to detect and classify one or more road signs and light signals from within live colour images captured by a camera. It attempts to develop an on-board warning system to alert the driver of the warning signs and light signals. Developing a reliable road sign and signal recognition system is considered a challenging task in compute vision.

2.Literature Survey

Sl no	Title of the paper	Authors	Year of publication	Dataset used	Methodology / Technology	Performance metrics	Drawbacks
1	A study on traffic sign recognition in scene image using genetic algorithms and neural networks	Y. Aoyagi, T. Asakura	22nd International Conference on Industrial Electronics, Control, and Instrumentation, IEEE, August (1996)	Traffic sign from a video image	Genetic algorithms and Neural networks	Among the 24 patterns other than the speed sign, it was recognized that only one pattern was not a sign	Image pattern recognition has been chiefly researched only for an individual object. However, it is an advanced direction to recognize the object which becomes a target from a scene image, with the development of the visual system of the robot
2	Real time traffic sign detection using color and shape-based features	TT le, ST Tran, S Mita, TD Nguyen	Asian Conference on Intelligent …, 2010 - Springer	12132 images from the eight cameras	Color and shape-based features	This proved toyield a speed of 2 fps at a level of 96% detection rate and 2 false	(i)The selective extraction of windows of interest, followed by their classification (ii) Exhaustive sliding window based classification

						positives per image	
3	Road sign detect ion and recognitio n using matching pursuit method	SH Hsu, CL Huang	Image and Vision Computing, 2001 - Elsevier	(a)30 triangular road signs; and (b) 10 circular road signs	Matching pursuit method	Triangular road signs:94% Circular road signs:91%	Edges are tested at different levels of resolution by using so- called a Hierarchical Structure Code. It is assumed that closed edgecontours are available at one of these levels of resolution, and failures happen when the outline of the traffic sign merges with the background.
4	Traffic Sign Detection and Recognitio n using Fully Convolutio nal Network	Yingyin g Zhu, Chengq uan Zhang, Duoyou Zhou, Xinggan g Wang, Xiang	2001	Swedish Traffic Signs Dataset (STSD)	Fully Convolutional Network Guided Proposals	average precision of 98.67%	color-based methods, shape-based methods and sliding window based methods.

	Guided Proposals	Bai, WenyuLiu,elsvier.					
5	Generalized traffic sign detection model for developing a sign inventory -	Y Tsai, P Kim, Z Wang	Journal of Computing in Civil Engineering, 2009	LADOTD roadway video log image sets.(37,640 video log images)	Neural networks	images with sign in them 83.67% images with no sign in them= 80.19%	proprietary algorithms use specific color filters and the features of specific shapes to distinguish a specific type of traffic sign.but they can detect only stop signs
6	Image segmentation and shape analysis for road-sign detection	JF Khan, SMA Bhuiyan	2011 - ieeexplore.ieee.org	121 different road-sign images	Genetic algorithms and Neural networks	88.88%	object detection has been image feature clustering. Bahlmann et al. [26] detected signs using a set of color-sensitive Haar wavelet features obtained from AdaBoost training and temporal information propagation.
7	Efficient algorithm for automatic road sign	C Souani, H Faiedh, K	Journal of real-time image processing,	The images were taken at different light conditions	Fully Convolutional Network Guided Proposals	The recognition rate achieved by the	First, the image is color segmented. Second, the

	recognition and its hardware implementation	Besbes	2014 - Springer			system was around 82 %	shape is extracted, and finally, the traffic sign is recognized by processing the local region.
8	Traffic sign recognition algorithm based on shape signature and dual-tree complex wavelet transform	Z Cai, M Gu	Journal of Central South University, 2013 - Springer	60 000 traffic sign images from several traffic video sequences	algorithm based on shape signature and dual-tree complex wavelet transform	95.39% at the peak	1) Edge detection in gray image. 2) Clustering and intelligent feature analysis. 3) Image segmentation by threshold in specific color space, then analysis with geometrical edge
9	Fast traffic sign recognition with a rotation invariant binary pattern based feature - Sensors,	S Yin, P Ouyang, L Liu, Y Guo, S Wei	2015 - mdpi.com	GTSRB and STS	Fully Convolutional Network Guided Proposals	1)HOG+ANN= 95.41% 2)SIFT+ANN=97.10 %	traffic sign detection and recognition can be divided into three categories. First, pre-processing methods are researched to locate and recognize the traffic signs. Second, pre-

							processing methods combining with classification are adopted to achieve robust traffic signs recognition. Third, specific design features combing with the classifiers are used to achieve the robust and computing efficient recognition.
10	Traffic sign recognition with hinge loss trained convolutional neural networks	J Jin, K Fu, C Zhang	IEEE Transactions on Intelligent …, 2014 - ieeexplore.ieee.org	12 630 test images	Convolutional neural networks	accuracy of 93.65%	Bayesian classifiers , boosting , tree classifiers , and support vector machines (SVMs) . These methods, from today's point of view, are considered using hand-coded features such as a circle detector in , a Haar wavelet [6] in , and a histogram of oriented gradient (HOG) or scale-invariant feature

							transform (SIFT)

3.Overview of the Work

3.1 Problem description

The objective of an automatic road sign and traffic signal recognition system is to detect and classify one or more road signs or light signals from within live colour images captured by a camera. It attempts to develop an on-board warning system to alert the driver of the warning signs by giving speech output.

Developing a reliable road sign and signal recognition system is considered a challenging task in computer vision.

3.2 Software Requirements

Python 3.7.2

Microsoft Excel 2016

Packages used

- Random
- Matplotlib
- Tensorflow
- Pandas
- Numpy

- Sci-kit
- Pickle

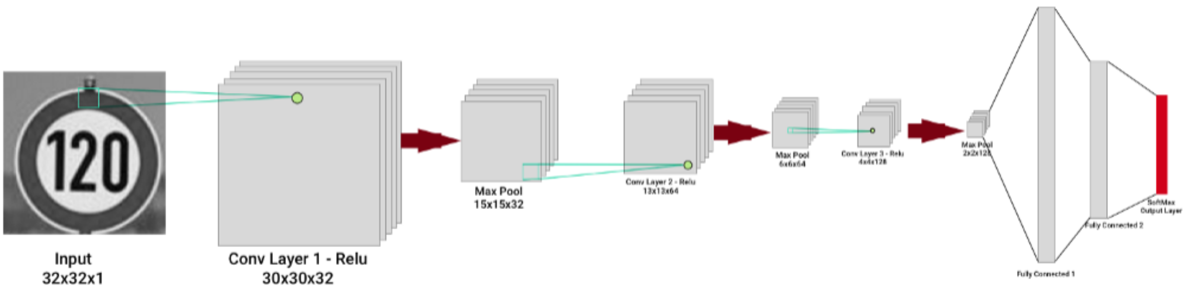
3.3 Hardware Requirements

- 400 MB Disk Space
- 8 GB RAM
- i7 Processer

4. System Design

4.1 The architecture proposed is inspired from Yann Le Cun's paper on classification of traffic signs. We added a few tweaks and created a modular codebase which allows us to try out different filter sizes, depth, and number of convolution layers, as well as the dimensions of fully connected layers. In homage to Le Cun, and with a touch of cheekiness, we called such network ***EdLeNet*** :).

We mainly tried 5x5 and 3x3 filter (aka kernel) sizes, and start with depth of 32 for our first convolutional layer. *EdLeNet*'s 3x3 architecture is shown below:



The network is composed of 3 convolutional layers—kernel size is 3x3, with depth doubling at next layer—using ReLU as the activation function, each followed by a 2x2 max pooling operation. The last 3 layers are fully connected, with the final layer producing 43 results (the total number of possible labels) computed using the SoftMax activation function. The network is trained using mini-batch stochastic gradient descent with the Adam optimizer. We build a highly modular coding infrastructure that enables us to *dynamically* create our models like in the following snippets:

The ModelConfig contains information about the model such as:

- The model function (e.g. EdLeNet)
- the model name
- input format (e.g. [32, 32, 1] for grayscale),
- convolutional layers config [filter size, start depth, number of layers],
- fully connected layers dimensions (e.g. [120, 84])
- number of classes
- dropout keep percentage values [p-conv, p-fc]

4.2 Design and Implementation Constraints

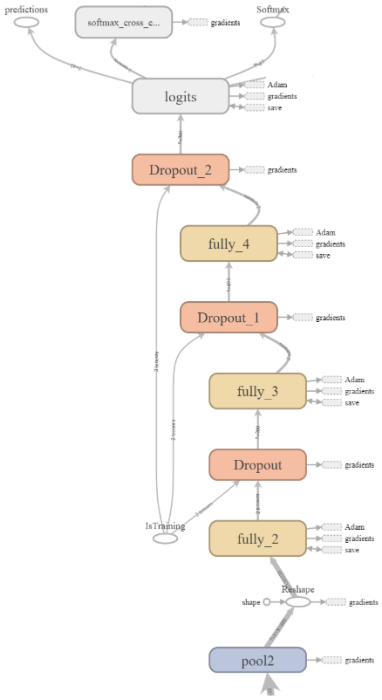
4.2.1 There are some areas that can create an issue for running the program. In this case, python libraries have been configured in this laptop. If another PC isn't configured according to the program it may cause problem.

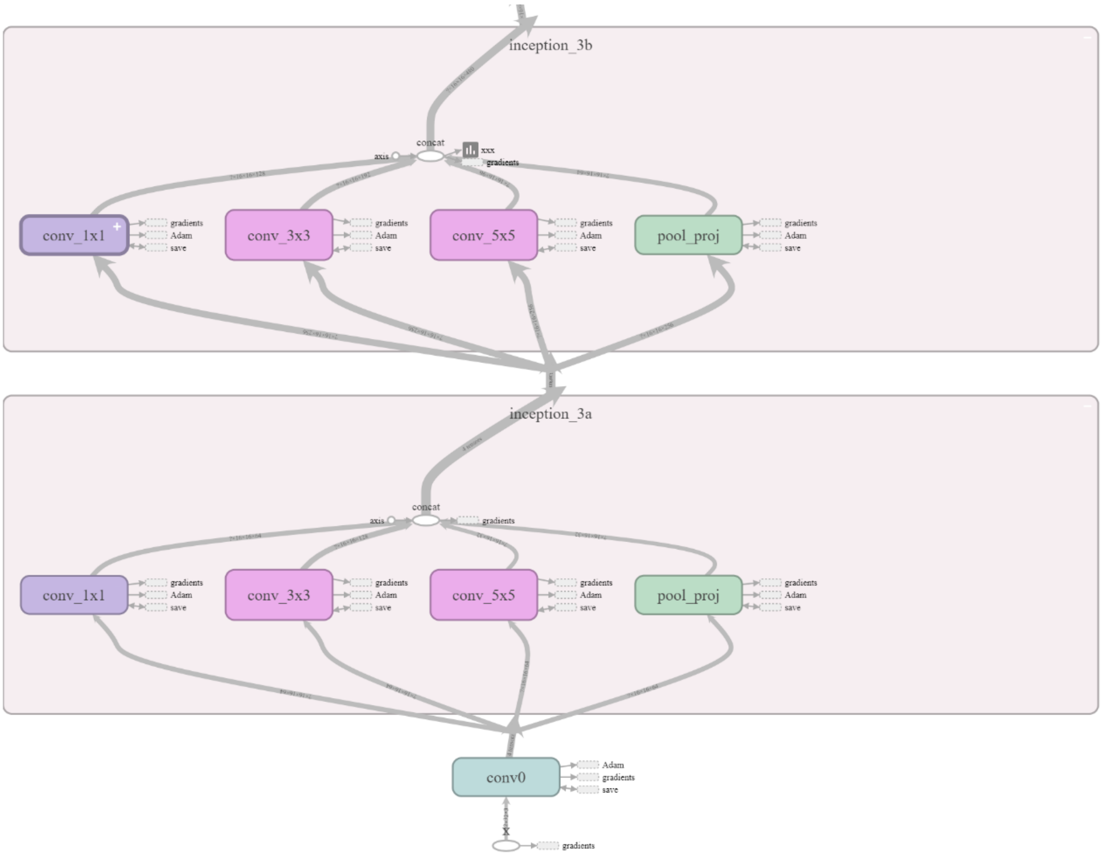
4.2.2 For Hardware there are no specific issues but processor may be the reason to effect the computation of the program

5.Implementation

5.1Description of Modules/Programs

FLOWCHART:





Step 0: Load The Data

Step 1: Dataset Summary & Exploration

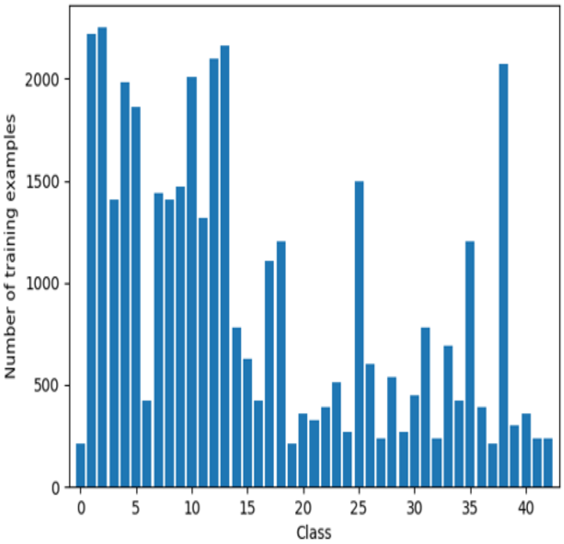
The pickled data is a dictionary with 4 key/value pairs:

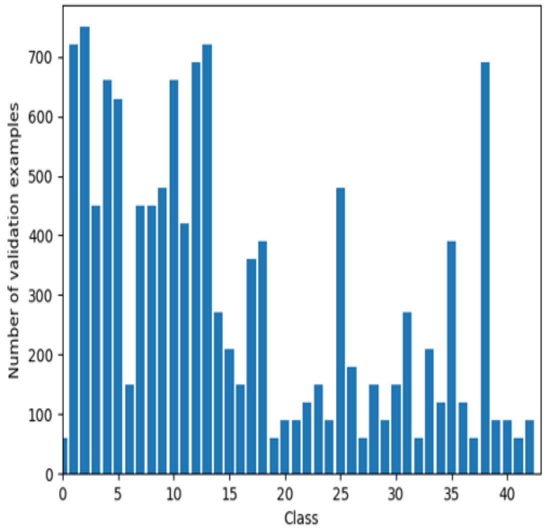
- 'features' is a 4D array containing raw pixel data of the traffic sign images, (num examples, width, height, channels).
- 'labels' is a 1D array containing the label/class id of the traffic sign. The file signnames.csv contains id -> name mappings for each id.
- 'sizes' is a list containing tuples, (width, height) representing the original width and height the image.
- 'coords' is a list containing tuples, (x1, y1, x2, y2) representing coordinates of a bounding box around the sign in the image. THESE COORDINATES ASSUME THE ORIGINAL

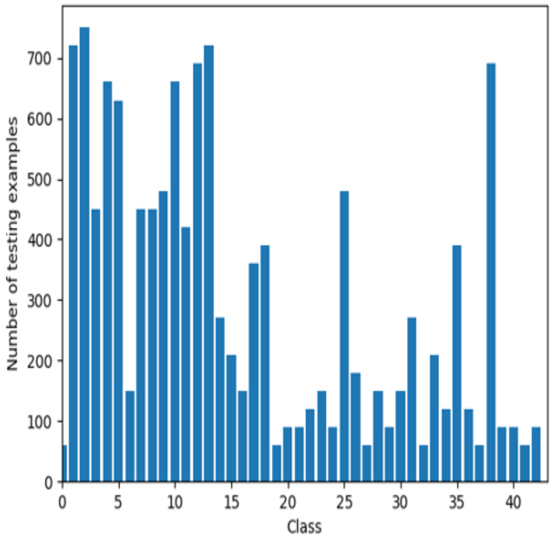
IMAGE. THE PICKLED DATA CONTAINS RESIZED VERSIONS (32 by 32) OF THESE IMAGES

Distribution

Now we are going to explore the distribution and take look at the distribution of classes in the training, validation and test set.







<u>Step 2: Design and Test a Model Architecture</u>

Design and implement a deep learning model that learns to recognize traffic signs. Train and test your model on the German Traffic Sign Dataset.

With the LeNet-5 solution, you should expect a validation set accuracy of about 0.89. To meet specifications, the validation set accuracy will need to be at least 0.93. It is possible to get an even higher accuracy.

There are various aspects to consider when thinking about this problem:

- Neural network architecture
- Play around preprocessing techniques (normalization, RGB to grayscale, etc.)
- Number of examples per label (some have more than others).
- Generate fake data.

Step 2.1:Pre-process the Data Set (normalization, grayscale, etc.)

- Minimally, the image data should be normalized so that the data has mean zero and equal variance. For image data, (pixel - 128)/ 128 is a quick way to approximately normalize the data and can be used in this project.
- Other pre-processing steps are optional. You can try different techniques to see if it improves performance.
- Use the code cell (or multiple code cells, if necessary) to implement the first step of your project.

Step 2.2:Data augmentation

The first thing we tried is to augment the data replicating the class labels which are rare in the dataset, so it can reduce ***high variance of our model***.

Note: We realized that data augmentation cannot make drastic improvements to the performance of our model, and the augmentation step was omitted due to slowing down the entire training procedure

Step 2.3 Train, Validate and Test the Model

A validation set can be used to assess how well the model is performing. A low accuracy on the training and validation sets imply under fitting. A high accuracy on the training set but low accuracy on the validation set implies overfitting.

Step 3: Test a Model on New Images

To give yourself more insight into how your model is working, download at least five pictures of German traffic signs and normal Light signals from the web and use your model to predict the traffic sign or the light signal type.

You may find signnames.csv useful as it contains mappings from the class id (integer) to the actual sign name and light name along with their meaning.

Output Top 5 Softmax Probabilities For Each Image Found on the Web

For each of the new images, print out the model's softmax probabilities to show the **certainty** of the model's predictions (limit the output to the top 5 probabilities for each image). tf.nn.top_k could prove helpful here.

The example below demonstrates how tf.nn.top_k can be used to find the top k predictions for each image.

tf.nn.top_k will return the values and indices (class ids) of the top k predictions. So if k=3, for each sign, it'll return the 3 largest probabilities (out of a possible 43) and the correspoding class ids.

Take this numpy array as an example. The values in the array represent predictions. The array contains softmax probabilities for five candidate images with six possible classes. tf.nn.top_k is used to choose the three classes with the highest probability:

(5, 6) array

a = np.array([[0.24879643, 0.07032244, 0.12641572, 0.34763842, 0.07893
497,

0.12789202],

[0.28086119, 0.27569815, 0.08594638, 0.0178669 , 0.18063401,

0.15899337],

[0.26076848, 0.23664738, 0.08020603, 0.07001922, 0.1134371 ,

0.23892179],

[0.11943333, 0.29198961, 0.02605103, 0.26234032, 0.1351348 ,

0.16505091],

[0.09561176, 0.34396535, 0.0643941 , 0.16240774, 0.24206137,

0.09155967]])

Running it through sess.run(tf.nn.top_k(tf.constant(a), k=3)) produces:

TopKV2(values=array([[0.34763842, 0.24879643, 0.12789202],

[0.28086119, 0.27569815, 0.18063401],

[0.26076848, 0.23892179, 0.23664738],

[0.29198961, 0.26234032, 0.16505091],

[0.34396535, 0.24206137, 0.16240774]]), indices=array([[3, 0, 5],

[0, 1, 4],

[0, 5, 1],

[1, 3, 5],

[1, 4, 3]], d type=int32))

Looking just at the first row we get [0.34763842, 0.24879643, 0.12789202], you can confirm these are the 3 largest probabilities in a. You'll also notice [3, 0, 5] are the corresponding indices.

Step 4 : Visualize the Neural Network's State with Test Images

While neural networks can be a great learning device they are often referred to as a black box. We can understand what the weights of a neural network look like better by plotting their feature maps. After successfully training your neural network you can see what it's feature maps look like by plotting the output of the network's weight layers in response to a test stimuli image. From these plotted feature maps, it's possible to see what characteristics of an image the network finds interesting. For a sign, maybe the inner network feature maps react with high activation to the sign's boundary outline or to the contrast in the sign's painted symbol.

5.2 Source Code

MODEL TRAINING

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import tensorflow as tf
from PIL import Image
import os
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from keras.models import Sequential, load_model
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout
from sklearn.metrics import accuracy_score
data = []
labels = []
classes = 46
cur_path = os.getcwd()
for i in range(classes):
    path = os.path.join(cur_path,'train',str(i))
    images = os.listdir(path)
    for a in images:
        try:
            image = Image.open(path + '\\'+ a)
            image = image.resize((30,30))
            image = np.array(image)
            data.append(image)
            labels.append(i)
        except:
            print("Error loading image")
data = np.array(data)
labels = np.array(labels)
print(data.shape, labels.shape)
```

```
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
y_train = to_categorical(y_train, 46)
y_test = to_categorical(y_test, 46)
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu', input_shape=X_train.shape[1:]))
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(46, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
epochs = 15
history = model.fit(X_train, y_train, batch_size=32, epochs=epochs, validation_data=(X_test, y_test))
y_test = pd.read_csv('Test.csv')
labels = y_test["ClassId"].values
imgs = y_test["Path"].values
data=[]
for img in imgs:
    image = Image.open(img)
    image = image.resize((30,30))
    data.append(np.array(image))
X_test=np.array(data)
pred = model.predict_classes(X_test)
from sklearn.metrics import accuracy_score
print(accuracy_score(labels, pred))
model.save('traffic_classifier.h5')
```

MAIN CODE

```
import tkinter as tk
from tkinter import filedialog
from tkinter import *
from PIL import ImageTk, Image
import numpy
import pyttsx3
engine = pyttsx3.init()
from keras.models import load_model
model = load_model('traffic_classifier.h5')
classes = { 1:'Speed limit (20km/h)',
            2:'Speed limit (30km/h)',
            3:'Speed limit (50km/h)',
            4:'Speed limit (60km/h)',
            5:'Speed limit (70km/h)',
            6:'Speed limit (80km/h)',
            7:'End of speed limit (80km/h)',
            8:'Speed limit (100km/h)',
            9:'Speed limit (120km/h)',
            10:'No passing',
            11:'No passing veh over 3.5 tons',
            12:'Right-of-way at intersection',
            13:'Priority road',
            14:'Yield',
            15:'Stop',
            16:'No vehicles',
            17:'Veh > 3.5 tons prohibited',
            18:'No entry',
            19:'General caution',
            20:'Dangerous curve left',
            21:'Dangerous curve right',
            22:'Double curve',
            23:'Bumpy road',
            24:'Slippery road',
            25:'Road narrows on the right',
```

```python
            26:'Road work',
            27:'Traffic signals',
            28:'Pedestrians',
            29:'Children crossing',
            30:'Bicycles crossing',
            31:'Beware of ice/snow',
            32:'Wild animals crossing',
            33:'End speed + passing limits',
            34:'Turn right ahead',
            35:'Turn left ahead',
            36:'Ahead only',
            37:'Go straight or right',
            38:'Go straight or left',
            39:'Keep right',
            40:'Keep left',
            41:'Roundabout mandatory',
            42:'End of no passing',
            43:'Other',
            44:'Red',
            45:'Yellow',
            46:'Green'}
top=tk.Tk()
top.geometry('800x600')
top.title('Traffic sign classification')
top.configure(background='#CDCDCD')
label=Label(top,background='#CDCDCD', font=('arial',15,'bold'))
sign_image = Label(top)
def classify(file_path):
    global label_packed
    image = Image.open(file_path)
    image = image.resize((30,30))
    image = numpy.expand_dims(image, axis=0)
    image = numpy.array(image)
    pred = model.predict_classes([image])[0]
    sign = classes[pred+1]
```

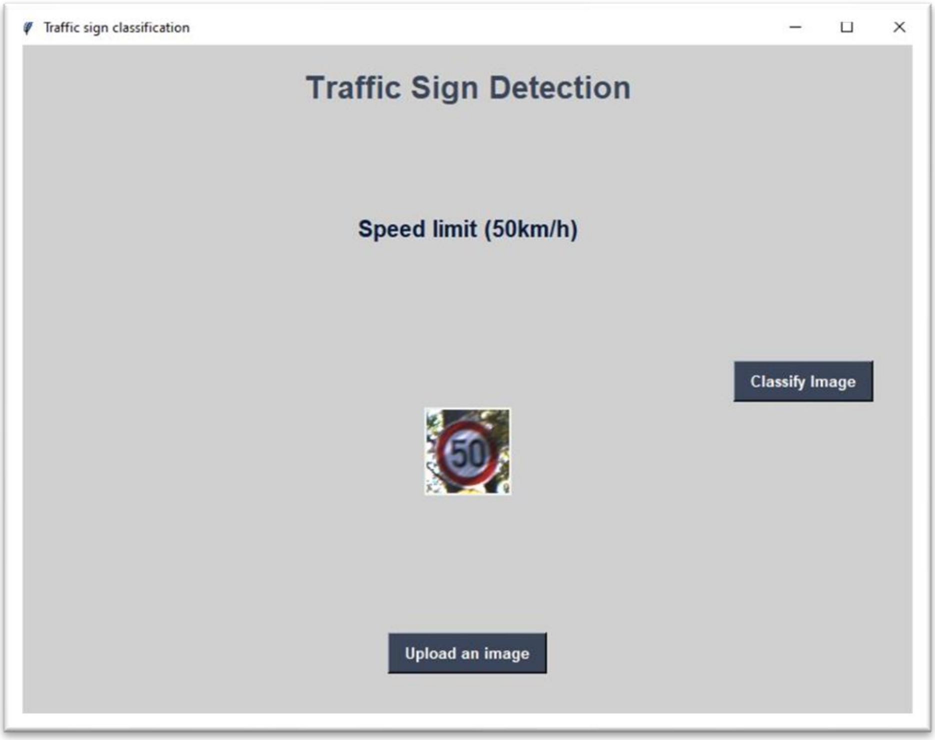
```
    if pred in range(43,46):
        print(sign+" Light")
        if pred == 43:
            engine.say("The traffic light is "+sign+" please stop your vehicle")
        elif pred == 44:
            engine.say("The traffic light is "+sign+" please prepare to leave")
        elif pred == 45:
            engine.say("The traffic light is "+sign+" you can move ahead")
        engine.runAndWait()
    else:
        print(sign)
        engine.say(sign)
        engine.runAndWait()
    label.configure(foreground='#011638', text=sign)
def show_classify_button(file_path):
    classify_b=Button(top,text="Classify Image",command=lambda: classify(file_path),padx=10,pady=5)
    classify_b.configure(background='#364156', foreground='white',font=('arial',10,'bold'))
    classify_b.place(relx=0.79,rely=0.46)
def upload_image():
    try:
        file_path=filedialog.askopenfilename()
        uploaded=Image.open(file_path)
        uploaded.thumbnail(((top.winfo_width()/2.25),(top.winfo_height()/2.25)))
        im=ImageTk.PhotoImage(uploaded)
        sign_image.configure(image=im)
        sign_image.image=im
        label.configure(text='')
        show_classify_button(file_path)
    except:
        pass
upload=Button(top,text="Upload an image",command=upload_image,padx=10,pady=5)
upload.configure(background='#364156', foreground='white',font=('arial',10,'bold'))
upload.pack(side=BOTTOM,pady=50)
sign_image.pack(side=BOTTOM,expand=True)
label.pack(side=BOTTOM,expand=True)
heading = Label(top, text="Traffic Sign Detection",pady=20, font=('arial',20,'bold'))
heading.configure(background='#CDCDCD',foreground='#364156')
heading.pack()
top.mainloop()
```

10.3 Execution snapshots

```
Using TensorFlow backend.
(40528, 30, 30, 3) (40528,)
(32422, 30, 30, 3) (8106, 30, 30, 3) (32422,) (8106,)
2020-11-03 18:37:56.783338: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX AVX2
Train on 32422 samples, validate on 8106 samples
Epoch 1/15
32422/32422 [==============================] - 63s 2ms/step - loss: 2.2861 - accuracy: 0.3967 - val_loss: 0.9657 - val_accuracy: 0.7351
Epoch 2/15
32422/32422 [==============================] - 68s 2ms/step - loss: 1.0614 - accuracy: 0.6860 - val_loss: 0.4315 - val_accuracy: 0.8756
Epoch 3/15
32422/32422 [==============================] - 61s 2ms/step - loss: 0.6760 - accuracy: 0.7940 - val_loss: 0.2619 - val_accuracy: 0.9297
Epoch 4/15
32422/32422 [==============================] - 68s 2ms/step - loss: 0.4299 - accuracy: 0.8732 - val_loss: 0.1418 - val_accuracy: 0.9613
Epoch 5/15
32422/32422 [==============================] - 61s 2ms/step - loss: 0.3404 - accuracy: 0.9023 - val_loss: 0.0907 - val_accuracy: 0.9782
Epoch 6/15
32422/32422 [==============================] - 62s 2ms/step - loss: 0.2893 - accuracy: 0.9160 - val_loss: 0.0994 - val_accuracy: 0.9697
Epoch 7/15
32422/32422 [==============================] - 72s 2ms/step - loss: 0.2761 - accuracy: 0.9233 - val_loss: 0.0743 - val_accuracy: 0.9796
Epoch 8/15
32422/32422 [==============================] - 69s 2ms/step - loss: 0.2522 - accuracy: 0.9306 - val_loss: 0.0747 - val_accuracy: 0.9785
Epoch 9/15
32422/32422 [==============================] - 61s 2ms/step - loss: 0.2594 - accuracy: 0.9300 - val_loss: 0.0655 - val_accuracy: 0.9826
Epoch 10/15
32422/32422 [==============================] - 70s 2ms/step - loss: 0.2318 - accuracy: 0.9382 - val_loss: 0.0538 - val_accuracy: 0.9853
Epoch 11/15
32422/32422 [==============================] - 73s 2ms/step - loss: 0.2379 - accuracy: 0.9364 - val_loss: 0.0887 - val_accuracy: 0.9747
Epoch 12/15
32422/32422 [==============================] - 70s 2ms/step - loss: 0.2288 - accuracy: 0.9411 - val_loss: 0.0495 - val_accuracy: 0.9868
Epoch 13/15
32422/32422 [==============================] - 63s 2ms/step - loss: 0.2463 - accuracy: 0.9349 - val_loss: 0.0691 - val_accuracy: 0.9820
Epoch 14/15
32422/32422 [==============================] - 64s 2ms/step - loss: 0.2404 - accuracy: 0.9380 - val_loss: 0.0791 - val_accuracy: 0.9788
Epoch 15/15
32422/32422 [==============================] - 61s 2ms/step - loss: 0.2564 - accuracy: 0.9377 - val_loss: 0.0542 - val_accuracy: 0.9859
0.9566112430720507
```


Traffic sign classification
Traffic Sign Detection
Green
Classify Image
Upload an image


Traffic sign classification
Traffic Sign Detection
Yellow
Classify Image
Upload an image


Traffic sign classification
Traffic Sign Detection
Speed limit (50km/h)
Classify Image
Upload an image


Traffic sign classification
Traffic Sign Detection
Turn right ahead
Classify Image
Upload an image

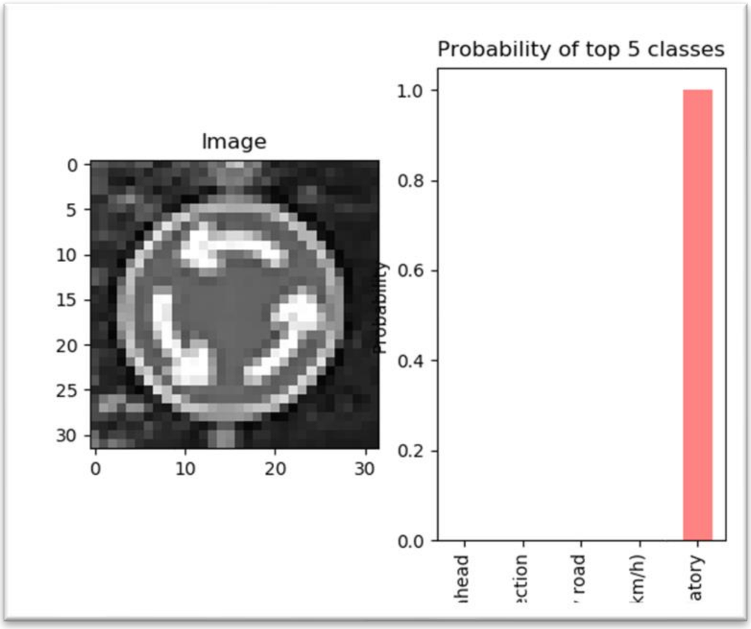


Image
Probability of top 5 classes
Probability
0
5
10
15
20
25
30
0
10
20
30
0.0
0.2
0.4
0.6
0.8
1.0
head
ction
road
km/h)
atory

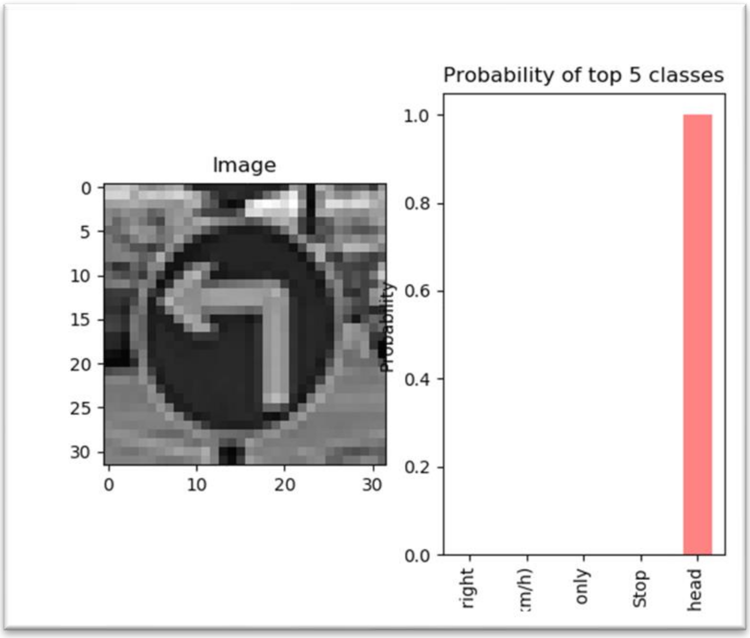


Image
Probability of top 5 classes
Probability
0
5
10
15
20
25
30
0
10
20
30
0.0
0.2
0.4
0.6
0.8
1.0
right
m/h)
only
Stop
head

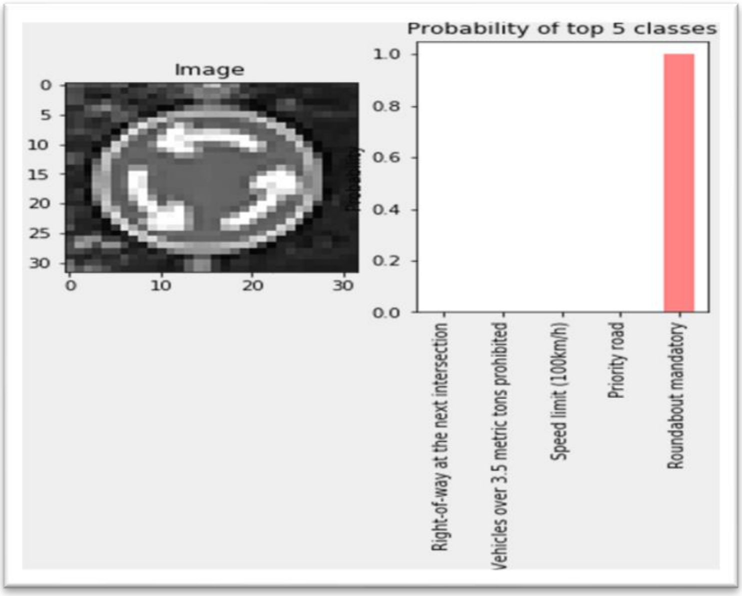


Image
Probability of top 5 classes
Probability
Right-of-way at the next intersection
Vehicles over 3.5 metric tons prohibited
Speed limit (100km/h)
Priority road
Roundabout mandatory

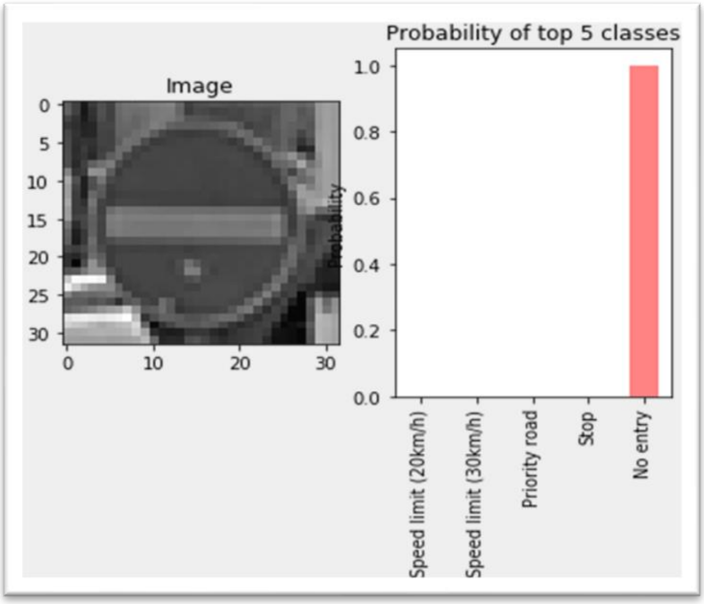


Image
Probability of top 5 classes
Probability
Speed limit (20km/h)
Speed limit (30km/h)
Priority road
Stop
No entry

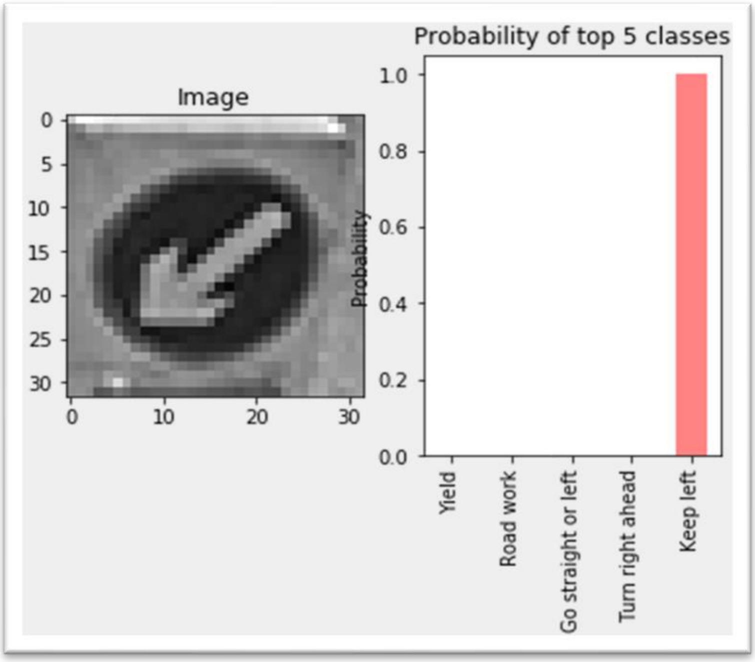


Image
Probability of top 5 classes
Probability
Yield
Road work
Go straight or left
Turn right ahead
Keep left

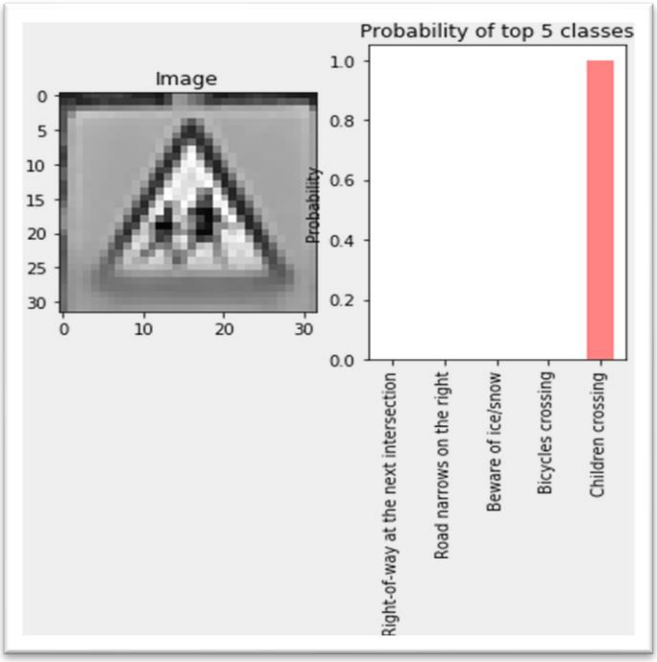
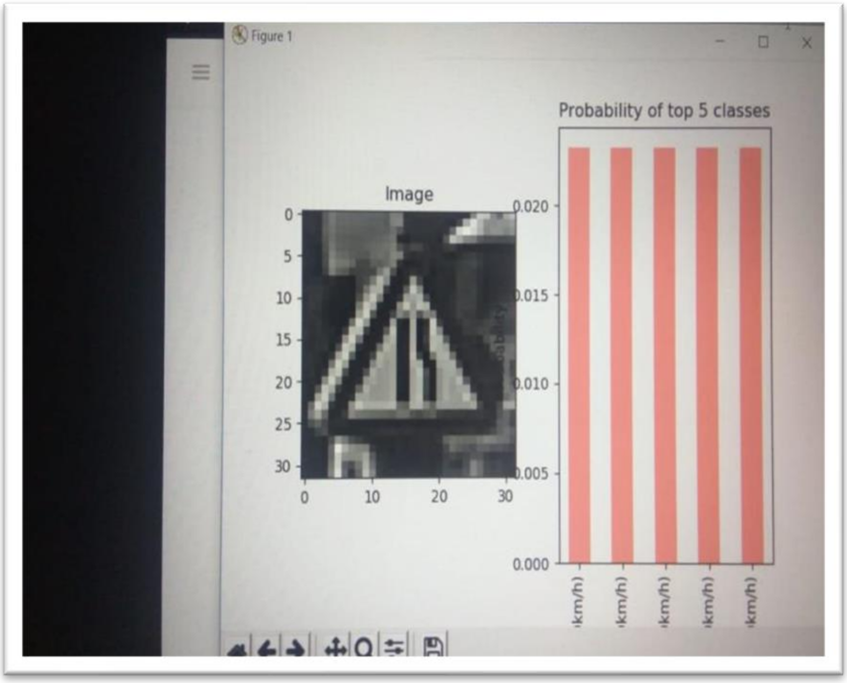
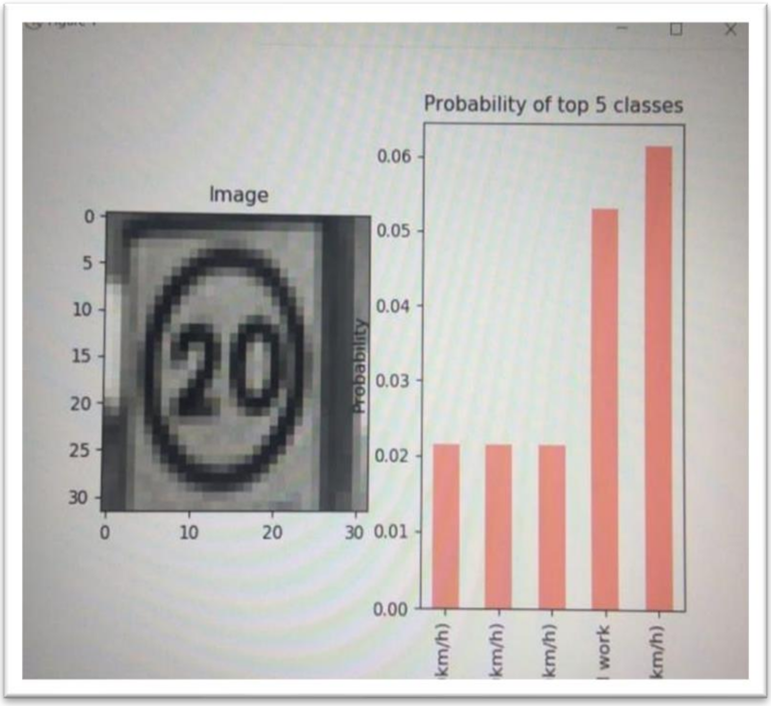


Image
Probability of top 5 classes
Probability
Right-of-way at the next intersection
Road narrows on the right
Beware of ice/snow
Bicycles crossing
Children crossing

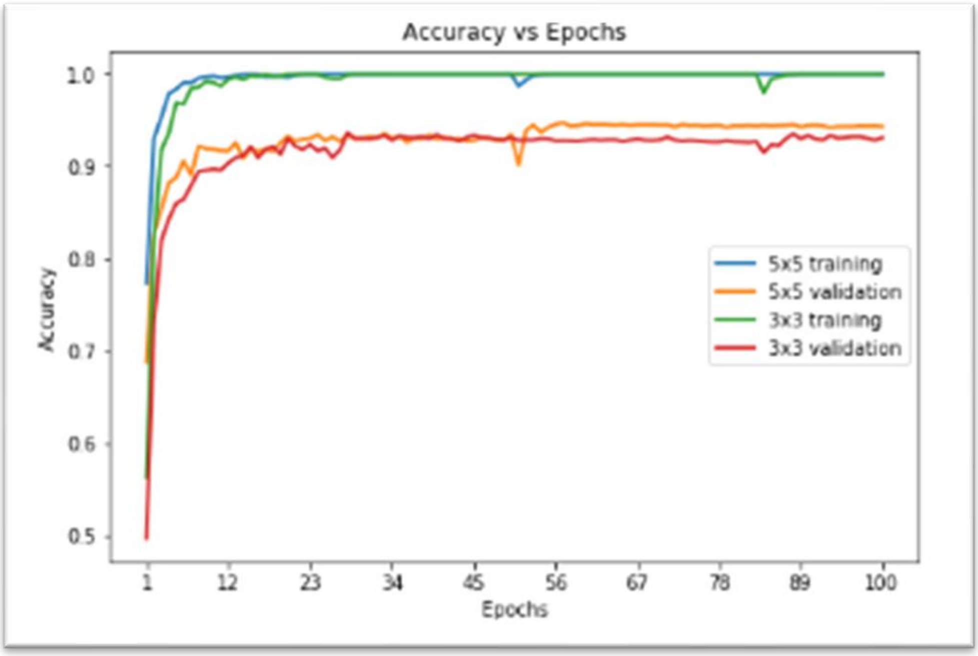
Wrong Detections:





6.Results:

Number of Training Images	49724
Number of Validation Images	10000
Number of Testing Images	10000
Number of Different Sign and Signals	46
Number of Detections	35



We probably don't have enough examples of such images in our test set for our model's predictions to improve. Additionally, while 91% test accuracy is very good

7. Conclusion and Future Directions

We covered how deep learning can be used to classify traffic signs and Traffic light signals with high accuracy, employing a variety of pre-processing and regularization techniques (e.g. dropout), and trying different model architectures. We built highly configurable code and developed a flexible way of evaluating multiple architectures. Our model reached close to close to 95% accuracy on the test set, achieving 98% on the validation set.

Exiting efficient traffic sign and traffic light signal detection method where locations of traffic signs are estimated together with their precise boundaries. To this end, generalized the object bounding box detection problem and formulated an object pose estimation problem, and the problem is effectively modeled using CNN based on the recent advances in object detection networks. The estimated pose of traffic signs is used to transform the boundary of traffic sign templates into the input image plane, providing precise boundaries with high accuracy. Since the base network is a dominant factor in speed-accuracy trade off, developing the base network specialized for traffic sign detection.

7.References

[1] Zeng, Yujun, et al. "Traffic sign recognition using deep convolutional networks and extreme learning machine." International Conference on Intelligent Science and Big Data Engineering. Springer, Cham, 2015.

[2] Jin, Junqi, Kun Fu, and Changshui Zhang. "Traffic sign recognition with hinge loss trained convolutional neural networks." IEEE Transactions on Intelligent Transportation Systems 15.5 (2014): 1991-2000.

[3] Neha Agrawal, Rahul Kumar Chaurasiya, Ensemble of SVM for Accurate Traffic Sign Detection and Recognition, Proceedings of the International Conference on Graphics and Signal Processing, June 24-27, 2017, Singapore, Singapore

[4] Sermanet, Pierre, and Yann LeCun. "Traffic sign recognition with multi-scale convolutional networks." Neural Networks (IJCNN), The 2011 International Joint Conference on. IEEE, 2011.

[5] De la Escalera, Arturo, J. Ma Armingol, and Mario Mata. " Traffic sign recognition and analysis for intelligent vehicles." Image and vision computing 21.3 (2003): 247-258.

[6] Garcia-Garrido, Miguel Angel, Miguel Angel Sotelo, and Ernesto Martin-Gorostiza.;Fast traffic sign detection and recognition under changing lighting conditions." Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE. IEEE, 2006.

[7] Sermanet, Pierre, and Yann LeCun. "Traffic sign recognition with multi-scale convolutional networks." Neural Networks (IJCNN), The 2011 International Joint Conference on. IEEE, 2011.

[8] Wu, Y., Liu, Y., Li, J., Liu, H., & Hu, X. (2013, August). Traffic sign detection based on convolutional neural networks. In Neural Networks (IJCNN), The 2013 International Joint Conference on (pp. 1-7). IEEE.

[9] Zhu, Y., Zhang, C., Zhou, D., Wang, X., Bai, X., & Liu, W. (2016). Traffic sign detection and recognition using fully convolutional network guided proposals. Neurocomputing, 214, 758-766.

[10] Zhu, Yingying, et al. "Traffic sign detection and recognition using fully convolutional network guided proposals; Neurocomputing 214 (2016): 758-766.