Traffic Sign Recognition System (Computer Vision and Machine learning)

Term Project

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

By:

SANGANA SAI VENKATA SURYA TEJA(EE23MTECH11032)

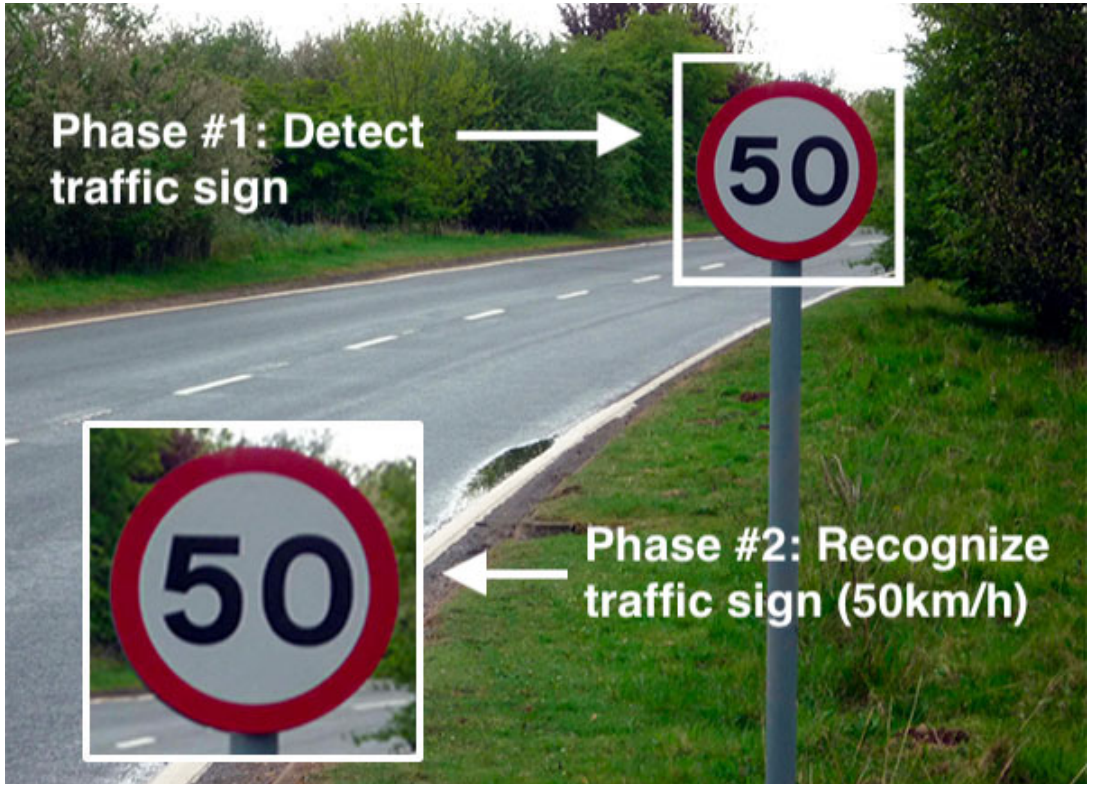
Contents:

Abstract (Motivation/Relevance of the project)

- In today's world, almost everything we do has been simplified by automated tasks. In an attempt to focus on the road while driving, drivers often miss out on signs on the side of the road, which could be dangerous for them and for the people around them.
- This problem can be avoided if there was an efficient way to notify the driver without having them to shift their focus. Traffic Sign Detection and Recognition (TSDR) plays an important role here by detecting and recognizing a sign, thus notifying the driver of any upcoming signs. This not only ensures road safety, but also allows the driver to be at little more ease while driving on tricky or new roads.
- With the help of this Advanced Driver Assistance Systems (ADAS) application, drivers will no longer face the problem of understanding what the sign says. we can implement a method for Traffic Sign Detection and Recognition using image processing for the detection of a sign and an ensemble of Convolutional Neural Networks (CNN) for the recognition of the sign. CNNs have a high recognition rate, thus making it desirable to use for implementing various computer vision tasks.

Specific objectives/goals of the project

This project will present a method for Traffic signs detection and classification. Its main objectives are, firstly, to recognize the meaning of the Traffic sign present in different images; secondly, to develop a method for traffic sign detection in a static image, or, preferably, in a video; lastly, to integrate both parts and provide real time traffic sign detection and identification.



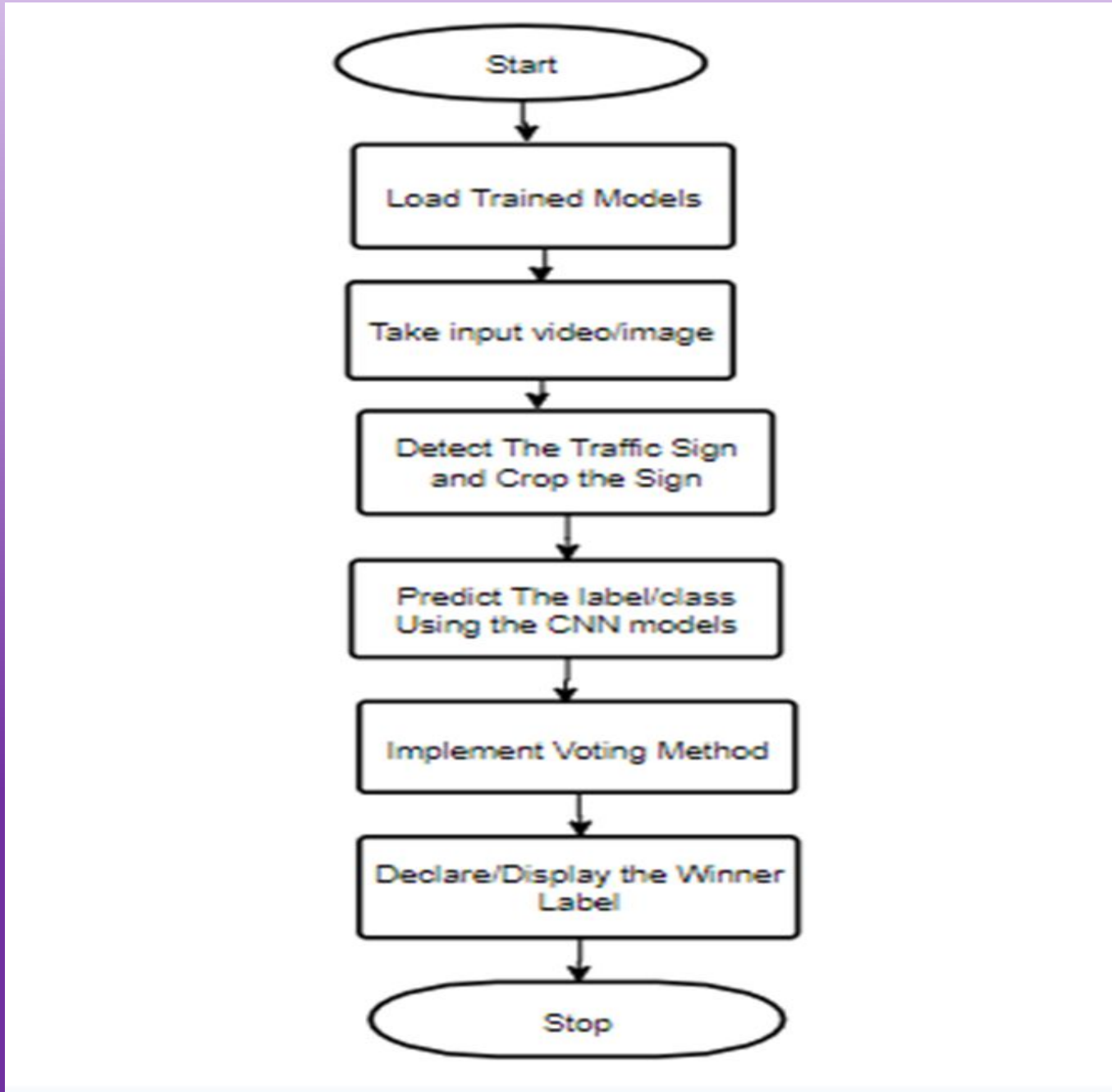
Dataset:

- German Traffic-Sign Dataset(GTSRB):-The German Traffic Sign Benchmark is a multi-class, single-image classification challenge held at the International Joint Conference on Neural Networks (IJCNN) 2011.

- The Image dataset consists of 43 classes (Unique traffic sign images).Training Set has 34799 Images , Test set has 12630 images and the validation set has 4410 images.

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

Flowchart:


Start
Load Trained Models
Take input video/image
Detect The Traffic Sign and Crop the Sign
Predict The label/class Using the CNN models
Implement Voting Method
Declare/Display the Winner Label
Stop

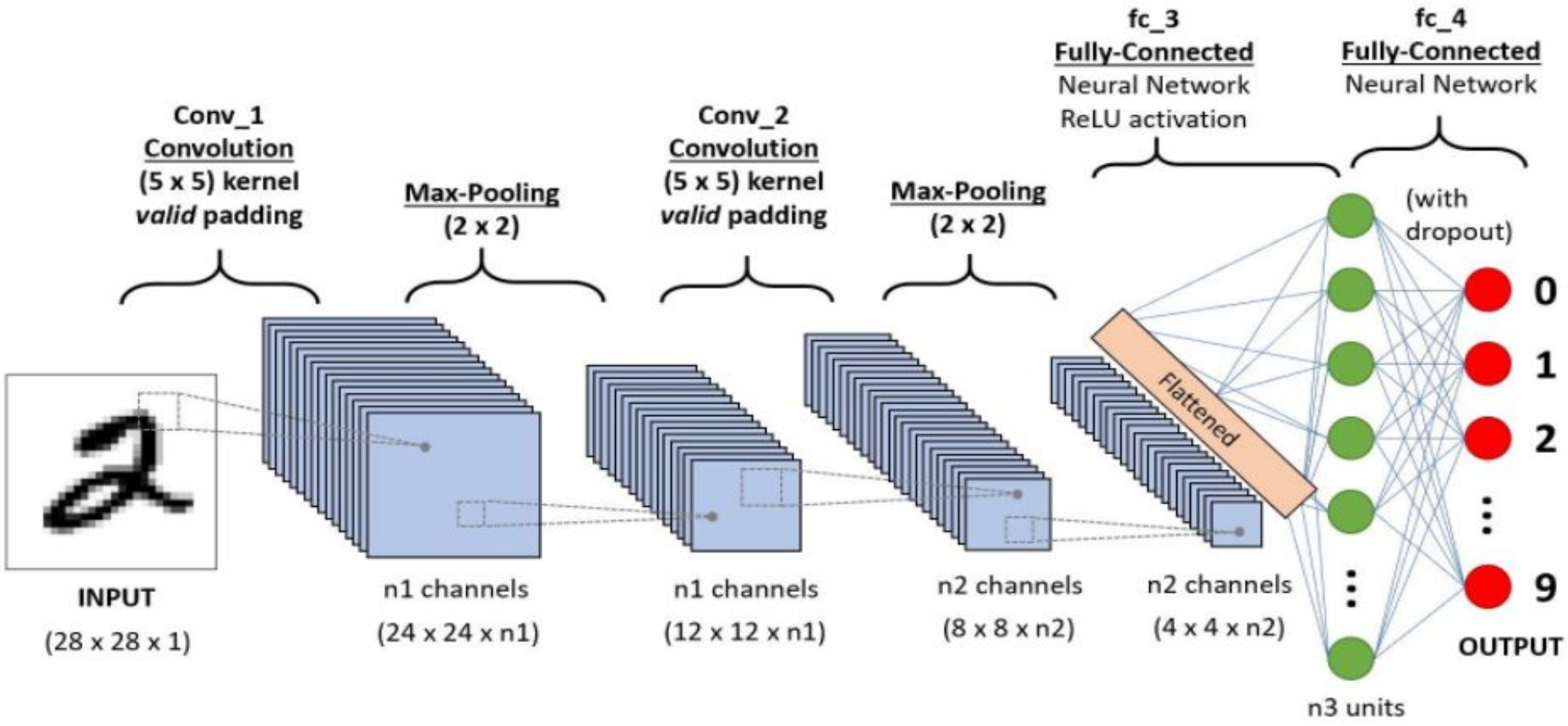
Reconciliation/Implementation Flow :

- **Will** use some image processing techniques to detect the traffic signs. The detection methods can be generally divided into color based, shape based and learning based methods.
- The output of the detection algorithm, which will be the input of the CNN, must resemble the images the CNN was trained on. That way, the accuracy should be as high as possible.
- When classification works satisfyingly enough for each model separately, voting should be implemented.

Convolutional Neural Network:

- 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.

Architecture of CNN:



Challenges anticipated :

- Building a Convolutional Neural Networks for classification of Traffic Signs.
- Optimizing the CNN model to achieve high accuracy while ensuring real-time performance is a challenge.
- Need to find a method for detecting/cropping the image of Traffic Sign from Full image of Road.

How do I expect to meet these challenges? (answer from the machine learning perceptive)

- I will attempt to construct the CNN with the necessary layers (i.e., convolution layer, activation layer, pooling layer, dropout layer, batch Normalisation Layer etc.).

- To enhance accuracy:
  > I will consider using a more effective loss function.
  > I will aim to use a superior activation function in the activation layer.
  > I will employ enhanced data augmentation techniques for training the dataset.
  > I will endeavor to construct multiple CNNs (perhaps 3) with varying layer configurations and will use voting to finalize the classification result.
  > I will investigate different optimization algorithms (like Adam, RMSprop, SGD with momentum) to determine which provides the best convergence and accuracy for the model.

- I will explore advanced image processing and edge detection techniques, such as Canny Edge Detection, to identify traffic signs.

Algorithm Description:

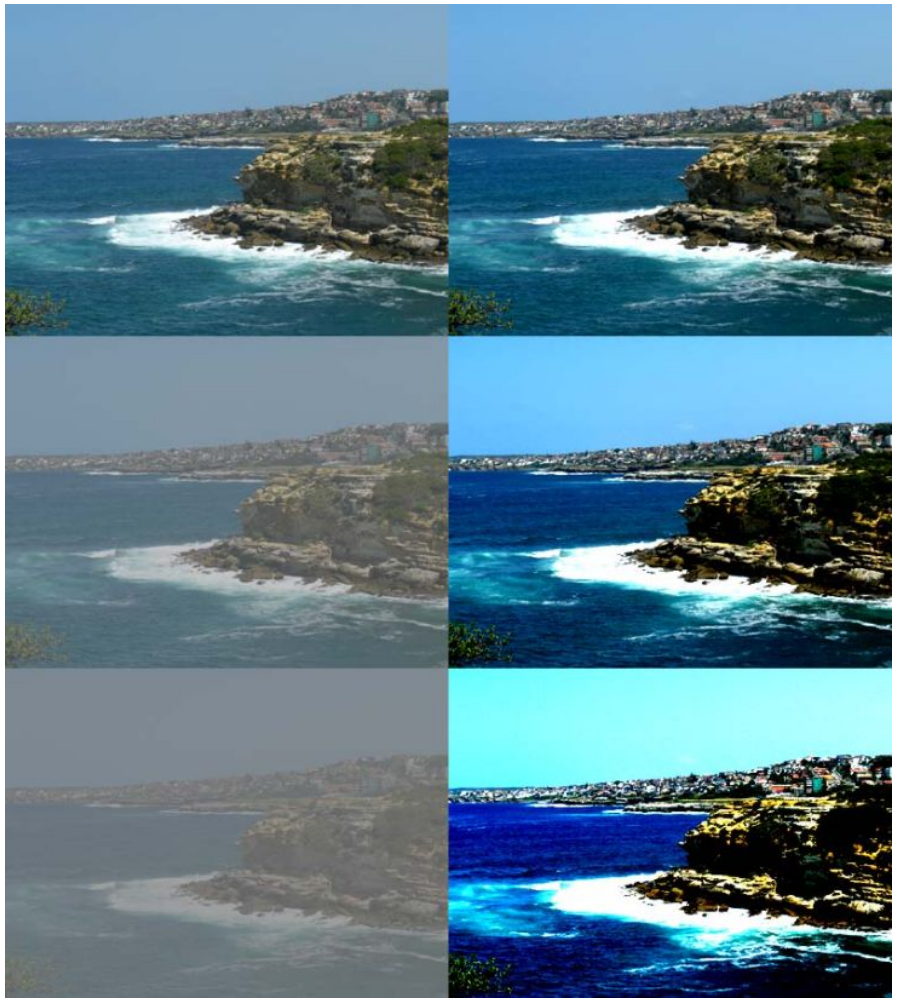
Traffic Sign Detection and Recognition(TSD) includes different algorithm steps which include :

- A)Contrast image
- B)Masking
- C)Canny edge detection
- D)Contour handling
- E)Classification
- F)Voting
- G)Reconciliation

Contrast image:

- **Contrast** is the difference in luminance or color that makes an object (or its representation in an image or display) distinguishable. In visual perception of the real world, contrast is determined by the difference in the color and brightness of the object and other objects within the same field of view.
- The human visual system is more sensitive to contrast than absolute luminance; we can perceive the world similarly regardless of the huge changes in illumination over the day or from place to place. The maximum *contrast* of an image is the contrast ratio or dynamic range.
- The image is first converted to YCrCb colorspace, consisting of Y - luminance or Luma, Cr - how far Luma is from the red component and Cb - how far Luma is from the blue component.
- Then the histogram of the Luma is equalized, spreading out the values from any cluster, and finally, the image is converted back to RGB.

Contrast images:



Masking image:

- Image masking is the process of separating an image from its background, either to cause the image to stand out on its own or to place the image over another background.
- In the old days of film stripping, it was done by cutting a physical "mask"--a sheet of material such as rubylith--in the shape of the image, and then projecting the image through it.
- The image is converted to HSV color space at first. Then 5 masks, meaning images where only black and a color ranging between some given values, are obtained . The masks correspond to red, white, blue and white.
- Two masks are obtained for the color red as, the HSV color space being a circular one, red’s range is present on the upper, as well as on the lower boundary. Finally, the masks are combined.

Masking image:

Canny Edge Detection:

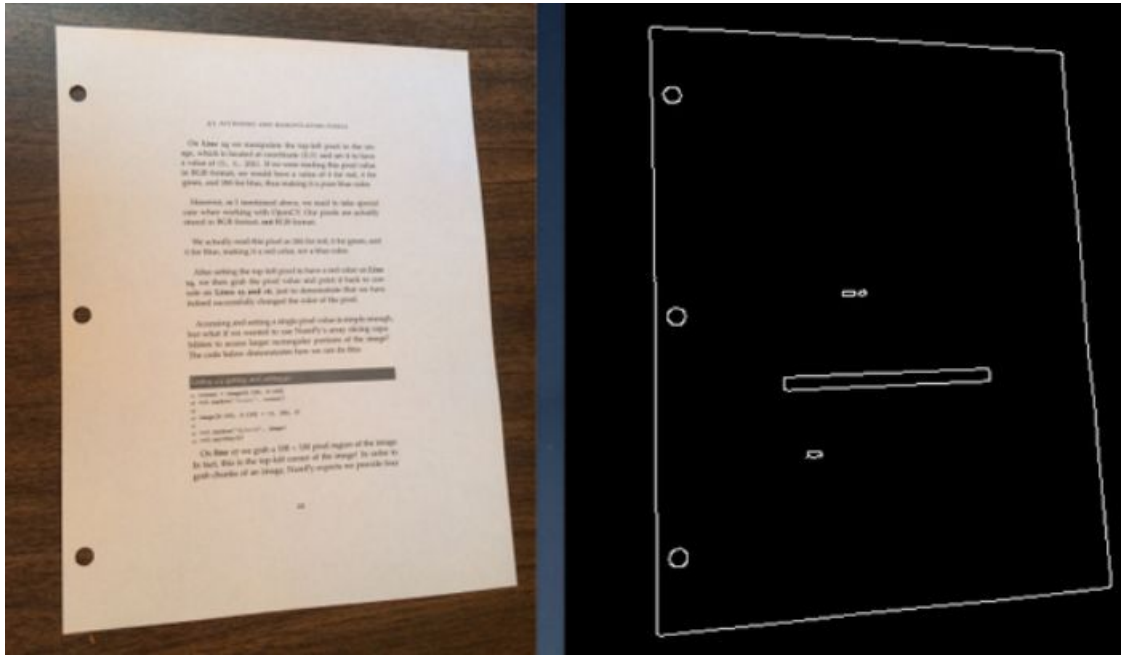
- The **Canny edge detector** is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. It was developed by John F. Canny in 1986. Canny also produced a *computational theory of edge detection* explaining why the technique works.
- Detection of edge with low error rate, which means that the detection should accurately catch as many edges shown in the image as possible
- The edge point detected from the operator should accurately localize on the center of the edge.
- A given edge in the image should only be marked once, and where possible, image noise should not create false edges.
- Then the intensity gradient is found for each pixel, all the pixels that are not local maximums being removed afterwards.
- Thresholding is then considered, based on two dynamic values, computed using the median of the image.
- Finally, only hard edges are obtained.
- The final part of preprocessing is converting the image to binary, leaving only black and white. After this, all the small components are removed by finding if their sizes, found through analyzing connected components, are below a given threshold

Canny edge detection:

Contour Handling:

- Contours handling is a process can be explained simply as a curve joining all the continuous points (along with the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition.
- Contours can be found easily on the obtained images by using the cv2 function called findContour().
- To find out if the contour really is a sign, its signature is compared to that of a circle.(The signature is found by dividing the distance from the center to each point of the contour to the maximum found distance, and averaging them.)
- If the threshold to which the value is compared is high enough, it will work for triangles and diamonds too, as all the signs are symmetric.
- Only the largest signs is cropped out of the image and passed on to classification.

Contour Handling:

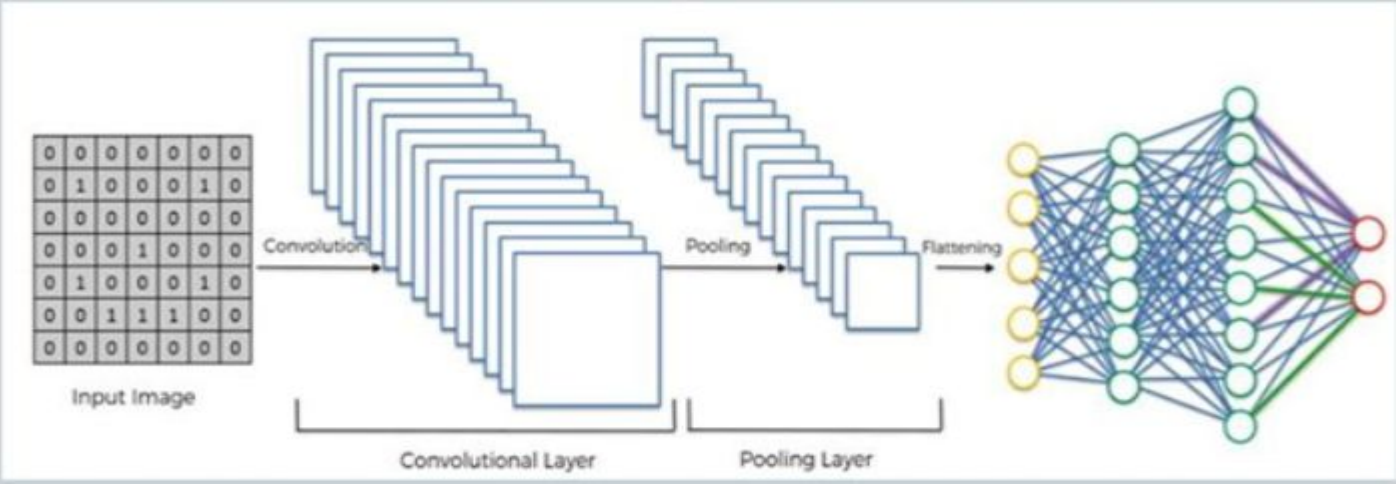


Classification:

- Various CNNs will be used for classification, but the dataset, training, validation and evaluation will be done in the same way, only the architecture of the neural network being different.
- The training data was split into three parts: training, with the most numerous images;
- Validation, used for computing the loss, a metric measuring how far the desired output is with respect to the actual output; and evaluation, which helps evaluate the trained model.
- When loading the images in memory, all of them are resized to 32x32 pixels, then local details are enhanced using an adaptive histogram and finally scaled between 0 and 1.
- The classes are weighted, as the numbers of images differ dramatically from one class to another.
- Weighting the classes prevents the model from certain classes taking over the probabilities.
- Augmenting the images by applying transformations like zooming, rotation, shifting and shearing prevents over fitting.

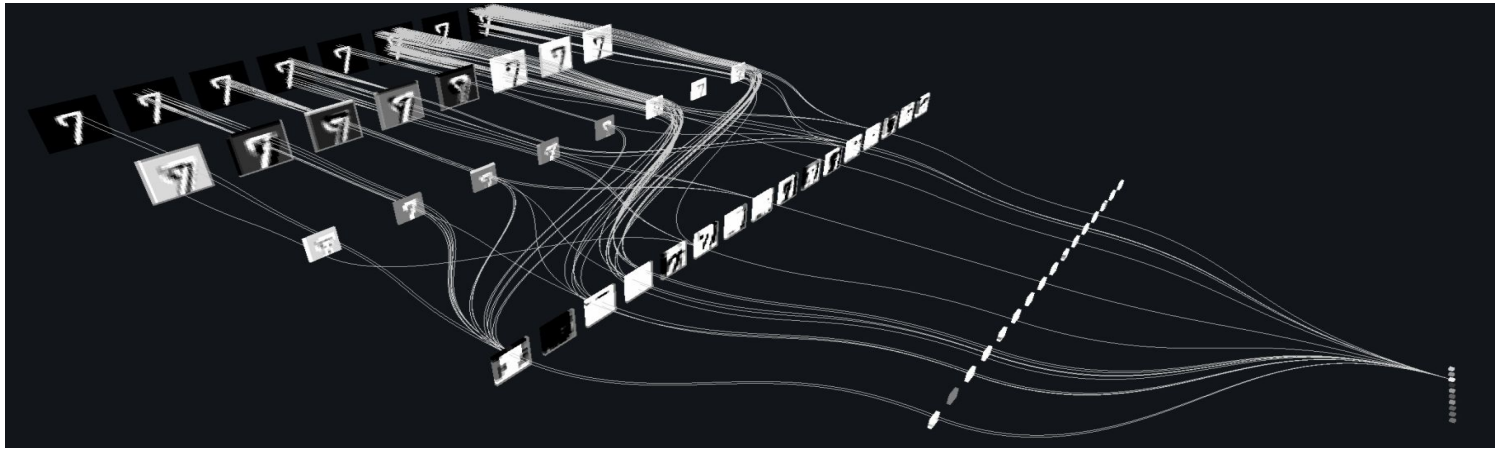
Voting:

- When trying to classify an input image, more CNN models will be used, each of them voting three labels, based on prediction accuracy.
- In order for the label to be voted, the accuracy should be greater than a given threshold, by default 90%.
- The label with the greatest sum is the winner, if there exists one.



Reconciliation:

- When put together, the two parts work well, as the majority of the erroneous data sent by the detector is rejected by the classifier.
- Voting certainly helps, as more models that behave differently can classify an image and come up with the best statistical result.



Architecture of CNN model-1

```
model = load_model("I:\downloads\model1")
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 6)         456
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 6)         0
_________________________________________________________________
activation (Activation)      (None, 14, 14, 6)         0
_________________________________________________________________
batch_normalization (BatchNo (None, 14, 14, 6)         24
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 10, 10, 66)        9966
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 66)          0
_________________________________________________________________
activation_1 (Activation)    (None, 5, 5, 66)          0
_________________________________________________________________
flatten (Flatten)            (None, 1650)              0
_________________________________________________________________
dropout (Dropout)            (None, 1650)              0
_________________________________________________________________
dense (Dense)                (None, 120)               198120
_________________________________________________________________
activation_2 (Activation)    (None, 120)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 120)               0
_________________________________________________________________
dense_1 (Dense)              (None, 84)                10164
_________________________________________________________________
activation_3 (Activation)    (None, 84)                0
_________________________________________________________________
dropout_2 (Dropout)          (None, 84)                0
_________________________________________________________________
dense_2 (Dense)              (None, 43)                3655
_________________________________________________________________
activation_4 (Activation)    (None, 43)                0
=================================================================
Total params: 222,385
Trainable params: 222,373
Non-trainable params: 12
```

Architecture of CNN model-2

```
model = load_model("I:\downloads\model2")
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 16)        448
_________________________________________________________________
batch_normalization (BatchNo (None, 32, 32, 16)        64
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 16)        2320
_________________________________________________________________
batch_normalization_1 (Batch (None, 32, 32, 16)        64
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 16)        0
_________________________________________________________________
dropout (Dropout)            (None, 16, 16, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 32)        4640
_________________________________________________________________
batch_normalization_2 (Batch (None, 16, 16, 32)        128
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 32)        9248
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 16, 32)        128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 32)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 8, 8, 32)          0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 64)          18496
_________________________________________________________________
batch_normalization_4 (Batch (None, 8, 8, 64)          256
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 64)          36928
_________________________________________________________________
batch_normalization_5 (Batch (None, 8, 8, 64)          256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 4, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 2048)              2099200
_________________________________________________________________
dropout_3 (Dropout)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              2098176
_________________________________________________________________
dropout_4 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               131200
_________________________________________________________________
dropout_5 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 43)                5547
=================================================================
Total params: 4,407,099
Trainable params: 4,406,651
Non-trainable params: 448
```

Architecture of CNN model-3

```
[7]: model = load_model("I:\downloads\model3")
     model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 8)         608
_________________________________________________________________
activation (Activation)      (None, 32, 32, 8)         0
_________________________________________________________________
batch_normalization (BatchNo (None, 32, 32, 8)         32
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 8)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 16, 16, 16)        1168
_________________________________________________________________
activation_1 (Activation)    (None, 16, 16, 16)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 16, 16, 16)        64
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 16)        2320
_________________________________________________________________
activation_2 (Activation)    (None, 16, 16, 16)        0
_________________________________________________________________
batch_normalization_2 (Batch (None, 16, 16, 16)        64
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 16)          0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 32)          4640
_________________________________________________________________
activation_3 (Activation)    (None, 8, 8, 32)          0
_________________________________________________________________
batch_normalization_3 (Batch (None, 8, 8, 32)          128
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 32)          9248
_________________________________________________________________
activation_4 (Activation)    (None, 8, 8, 32)          0
_________________________________________________________________
batch_normalization_4 (Batch (None, 8, 8, 32)          128
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 128)               65664
_________________________________________________________________
activation_5 (Activation)    (None, 128)               0
_________________________________________________________________
batch_normalization_5 (Batch (None, 128)               512
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
flatten_1 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               16512
_________________________________________________________________
activation_6 (Activation)    (None, 128)               0
_________________________________________________________________
batch_normalization_6 (Batch (None, 128)               512
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 43)                5547
_________________________________________________________________
activation_7 (Activation)    (None, 43)                0
=================================================================
Total params: 107,147
Trainable params: 106,427
Non-trainable params: 720
```

TRAINING AND TESTING FLOWCHART:

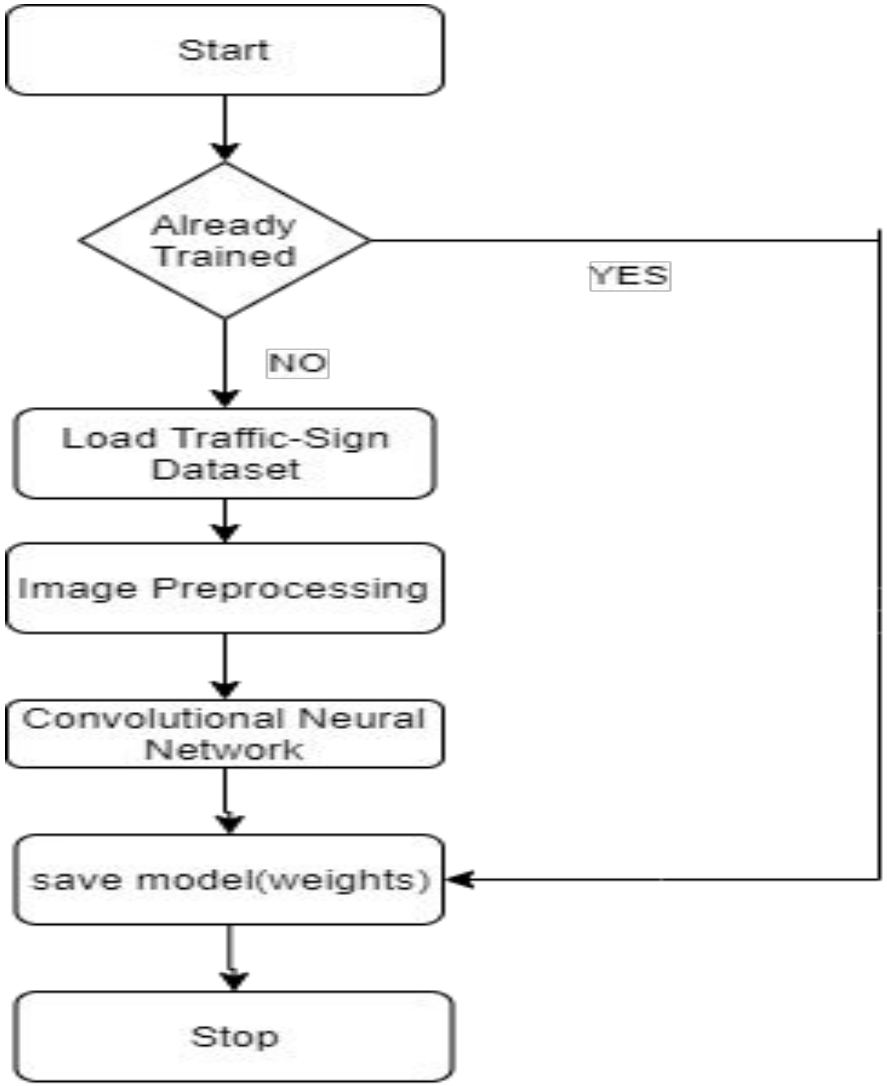


Fig: Training flowchart

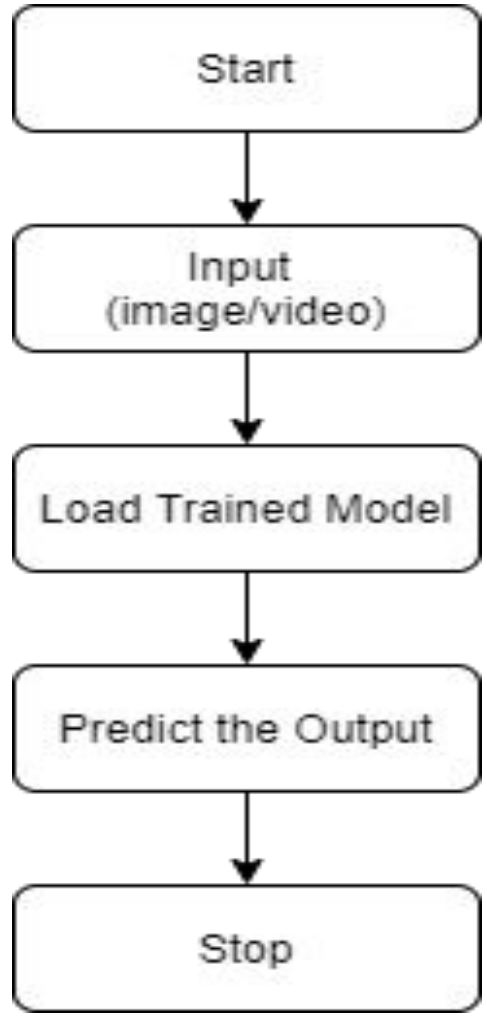


Fig: Testing Flowchart

Deep Convolutional Neural Network(Supervised Learning) :

Step 1: Initialize Hyperparameters (Learning rate , Batch size , No of Epochs)

Step 2:We initialize all filters and weights with random values.

Step 3:Compile the model with required optimizer(Adam),Loss(binary cross_entropy)and the metrics(Accuracy) to be calculated.

Step 4 : The training image is input to the network and goes over the forward propagation phases (i.e. convolution layer, ReLU layer , pooling layer,batch normalization layer,drop layer,dense layer actions along with forward propagation in the Fully Connected layer) and gives output probabilities for all class.

Step 5: Calculating the error(binary cross_entropy) at the output layer is given as Total Error

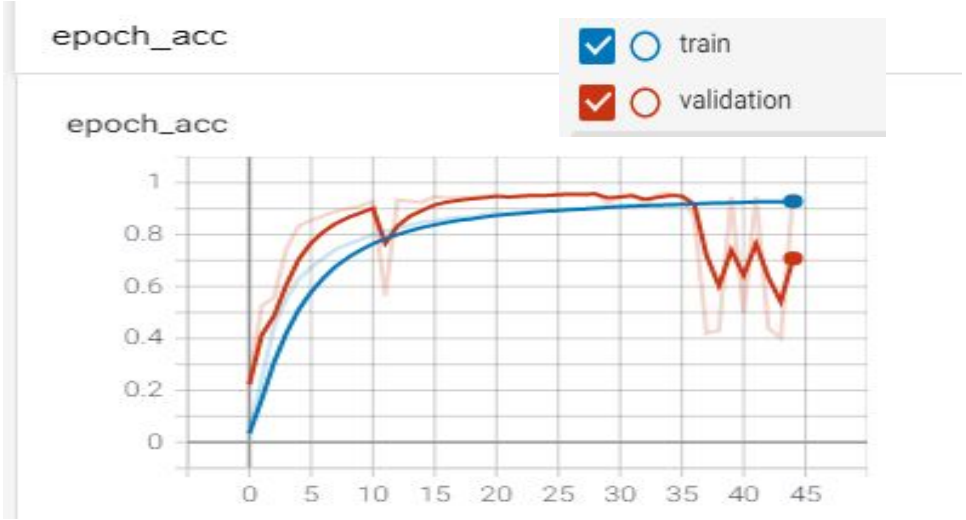
$$\text{Loss} = -\frac{1}{\text{output size}} \sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

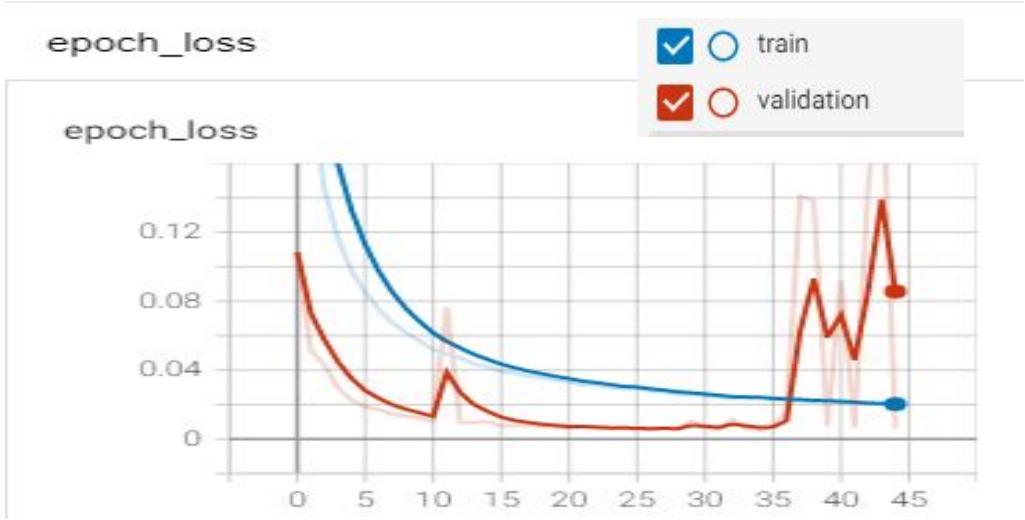
Step 6: Using Back propagation we compute the gradients of the error for all weights in the network and use gradient descent to update all filter values / weights and parameter values to minimize the output error. The weights are updated in proportion to put their influence to reducing the error.

CNN Model-1 CLASSIFICATION REPORT

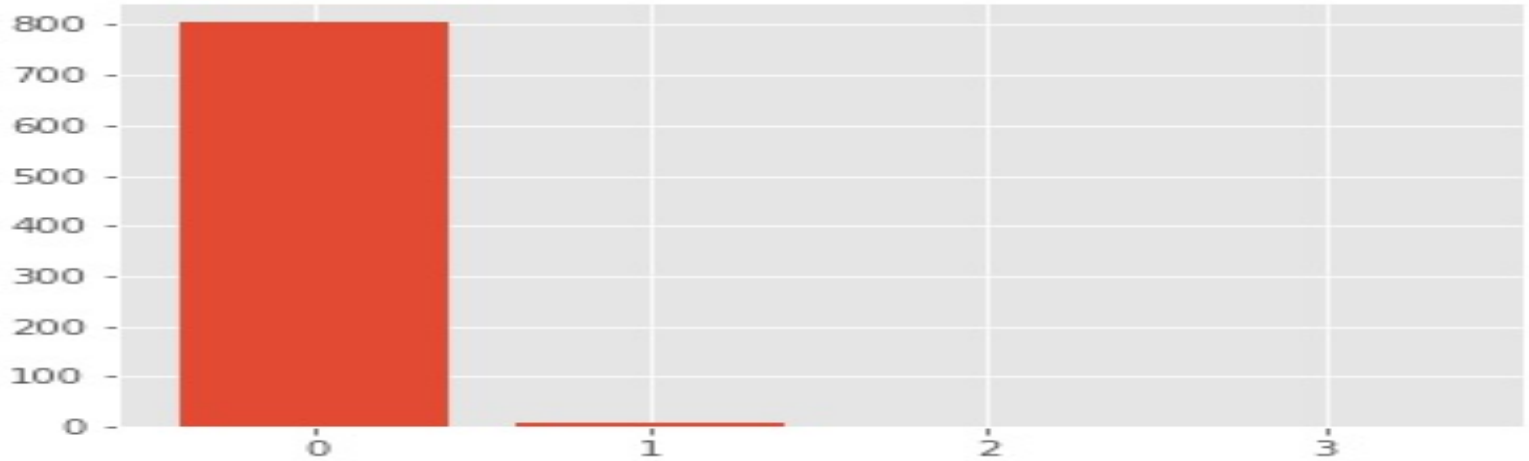
	precision	recall	f1-score	support
Speed limit (20km/h)	1.00	1.00	1.00	19
Speed limit (30km/h)	1.00	1.00	1.00	19
Speed limit (50km/h)	1.00	0.89	0.94	19
Speed limit (60km/h)	1.00	1.00	1.00	19
Speed limit (70km/h)	1.00	1.00	1.00	19
Speed limit (80km/h)	0.95	0.95	0.95	19
End of speed limit (80km/h)	0.95	1.00	0.97	19
Speed limit (100km/h)	0.94	0.84	0.89	19
Speed limit (120km/h)	0.90	1.00	0.95	19
No passing	0.95	0.95	0.95	19
No passing for vehicles over 3.5 metric tons	0.95	0.95	0.95	19
Right-of-way at the next intersection	0.95	1.00	0.97	19
Priority road	1.00	1.00	1.00	19
Yield	1.00	1.00	1.00	19
Stop	1.00	1.00	1.00	19
No vehicles	0.95	1.00	0.97	19
Vehicles over 3.5 metric tons prohibited	1.00	1.00	1.00	19
No entry	1.00	1.00	1.00	19
General caution	1.00	1.00	1.00	19
Dangerous curve to the left	1.00	1.00	1.00	19
Dangerous curve to the right	1.00	1.00	1.00	19
Double curve	1.00	1.00	1.00	19
Bumpy road	1.00	1.00	1.00	19
Slippery road	1.00	1.00	1.00	19
Road narrows on the right	0.95	1.00	0.97	19
Road work	1.00	1.00	1.00	19
Traffic signals	1.00	1.00	1.00	19
Pedestrians	1.00	1.00	1.00	19
Children crossing	1.00	1.00	1.00	19
Bicycles crossing	1.00	1.00	1.00	19
Beware of ice/snow	1.00	0.95	0.97	19
Wild animals crossing	1.00	0.95	0.97	19
End of all speed and passing limits	1.00	0.95	0.97	19
Turn right ahead	1.00	1.00	1.00	19
Turn left ahead	1.00	1.00	1.00	19
Ahead only	1.00	1.00	1.00	19
Go straight or right	1.00	1.00	1.00	19
Go straight or left	1.00	1.00	1.00	19
Keep right	1.00	1.00	1.00	19
Keep left	1.00	1.00	1.00	19
Roundabout mandatory	0.95	1.00	0.97	19
End of no passing	1.00	1.00	1.00	19
End of no passing by vehicles over 3.5 metric tons	1.00	1.00	1.00	19
accuracy			0.99	817
macro avg	0.99	0.99	0.99	817
weighted avg	0.99	0.99	0.99	817

Plot of Accuracy and Loss against the epochs During Training and Validation





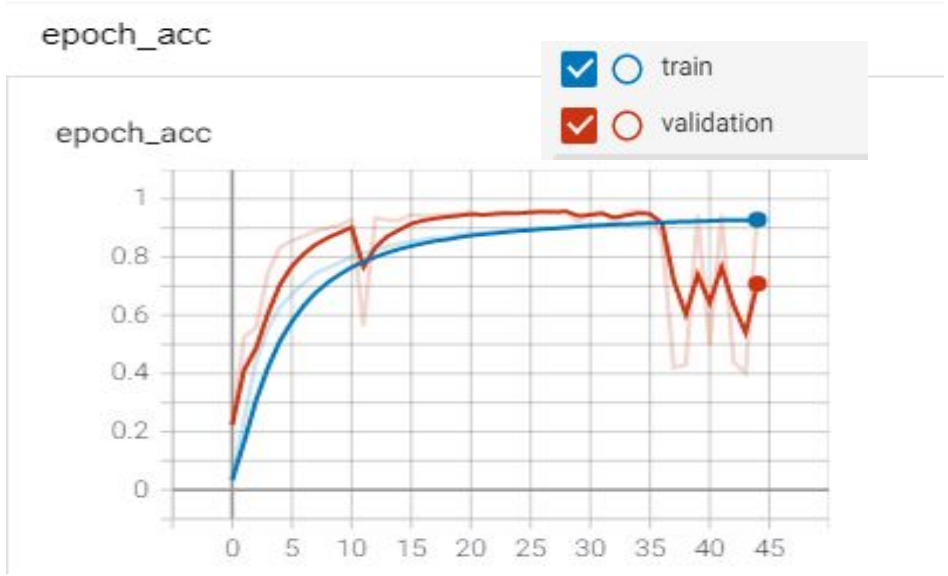
Evaluation total statistics (0 - recognized first choice; 1 - second choice; 2 - third choice; 3 - others [unrecognized])



CNN Model-2 CLASSIFICATION REPORT

	precision	recall	f1-score	support
Speed limit (20km/h)	0.95	1.00	0.97	19
Speed limit (30km/h)	1.00	0.95	0.97	19
Speed limit (50km/h)	1.00	0.95	0.97	19
Speed limit (60km/h)	0.95	1.00	0.97	19
Speed limit (70km/h)	1.00	1.00	1.00	19
Speed limit (80km/h)	0.95	0.95	0.95	19
End of speed limit (80km/h)	1.00	1.00	1.00	19
Speed limit (100km/h)	0.95	1.00	0.97	19
Speed limit (120km/h)	1.00	0.95	0.97	19
No passing	1.00	1.00	1.00	19
No passing for vehicles over 3.5 metric tons	1.00	1.00	1.00	19
Right-of-way at the next intersection	1.00	1.00	1.00	19
Priority road	1.00	1.00	1.00	19
Yield	1.00	1.00	1.00	19
Stop	1.00	1.00	1.00	19
No vehicles	1.00	1.00	1.00	19
Vehicles over 3.5 metric tons prohibited	1.00	1.00	1.00	19
No entry	1.00	1.00	1.00	19
General caution	1.00	1.00	1.00	19
Dangerous curve to the left	1.00	1.00	1.00	19
Dangerous curve to the right	1.00	1.00	1.00	19
Double curve	1.00	1.00	1.00	19
Bumpy road	1.00	1.00	1.00	19
Slippery road	1.00	1.00	1.00	19
Road narrows on the right	1.00	1.00	1.00	19
Road work	1.00	1.00	1.00	19
Traffic signals	1.00	1.00	1.00	19
Pedestrians	1.00	1.00	1.00	19
Children crossing	1.00	1.00	1.00	19
Bicycles crossing	1.00	1.00	1.00	19
Beware of ice/snow	1.00	1.00	1.00	19
Wild animals crossing	1.00	1.00	1.00	19
End of all speed and passing limits	1.00	1.00	1.00	19
Turn right ahead	1.00	1.00	1.00	19
Turn left ahead	1.00	1.00	1.00	19
Ahead only	1.00	1.00	1.00	19
Go straight or right	1.00	1.00	1.00	19
Go straight or left	1.00	1.00	1.00	19
Keep right	1.00	1.00	1.00	19
Keep left	1.00	1.00	1.00	19
Roundabout mandatory	1.00	1.00	1.00	19
End of no passing	1.00	1.00	1.00	19
End of no passing by vehicles over 3.5 metric tons	1.00	1.00	1.00	19
accuracy			1.00	817
macro avg	1.00	1.00	1.00	817
weighted avg	1.00	1.00	1.00	817

Fig:- Plot of Accuracy and Loss against the epochs During Training and Validation



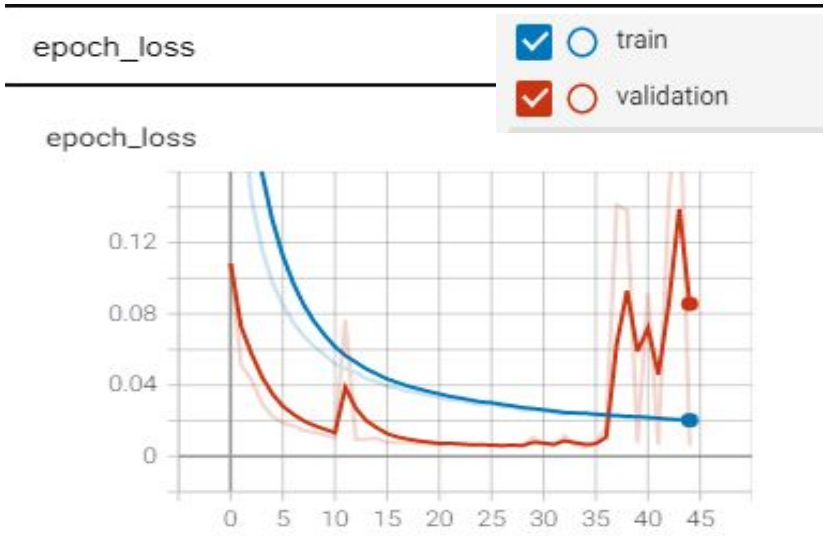
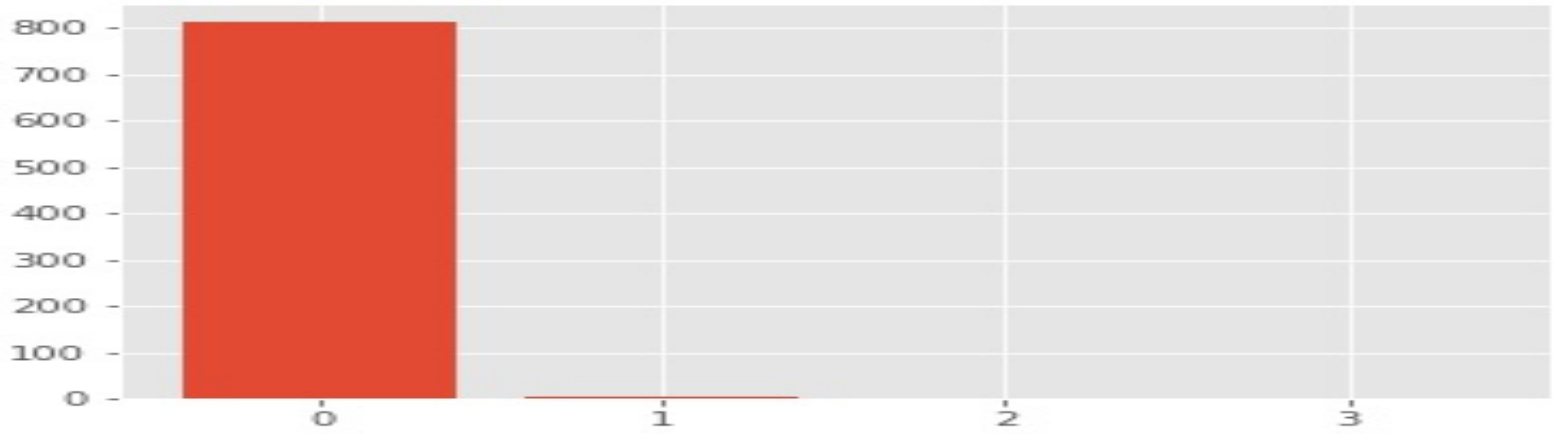


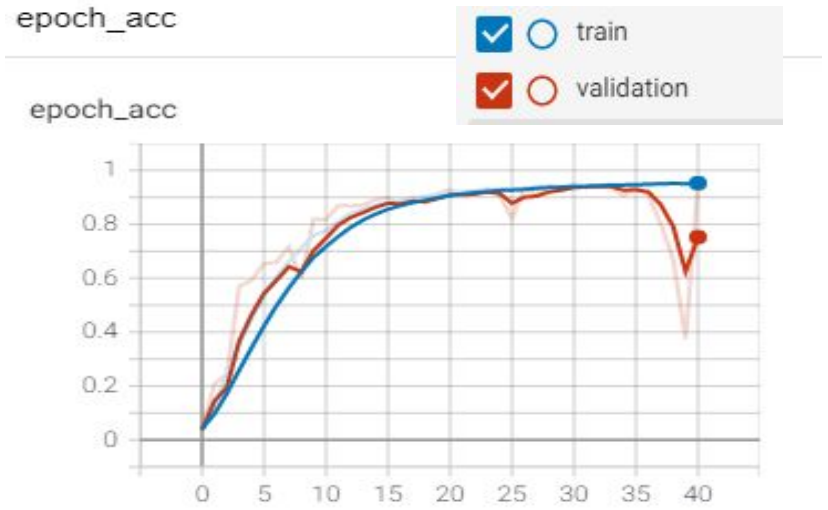
Fig:- Evaluation total statistics (0 - recognized first choice; 1 - second choice; 2 - third choice; 3 - others [unrecognized])

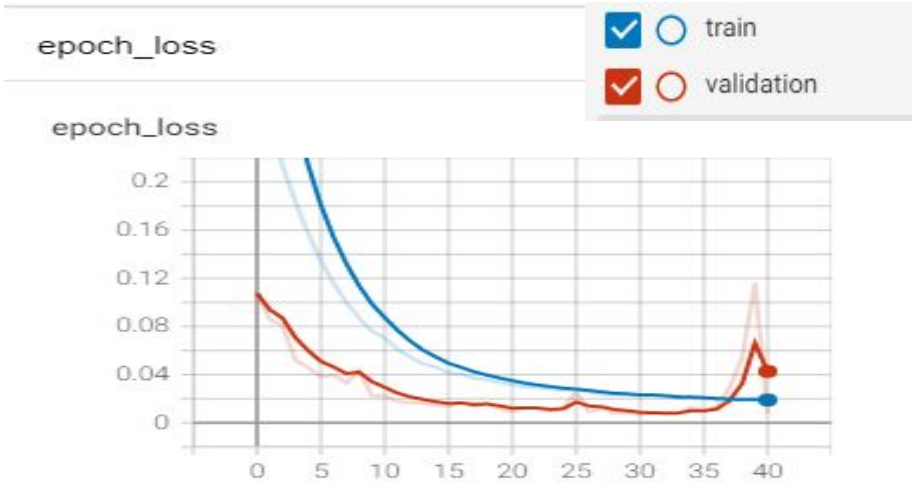


CNN Model-3 CLASSIFICATION REPORT

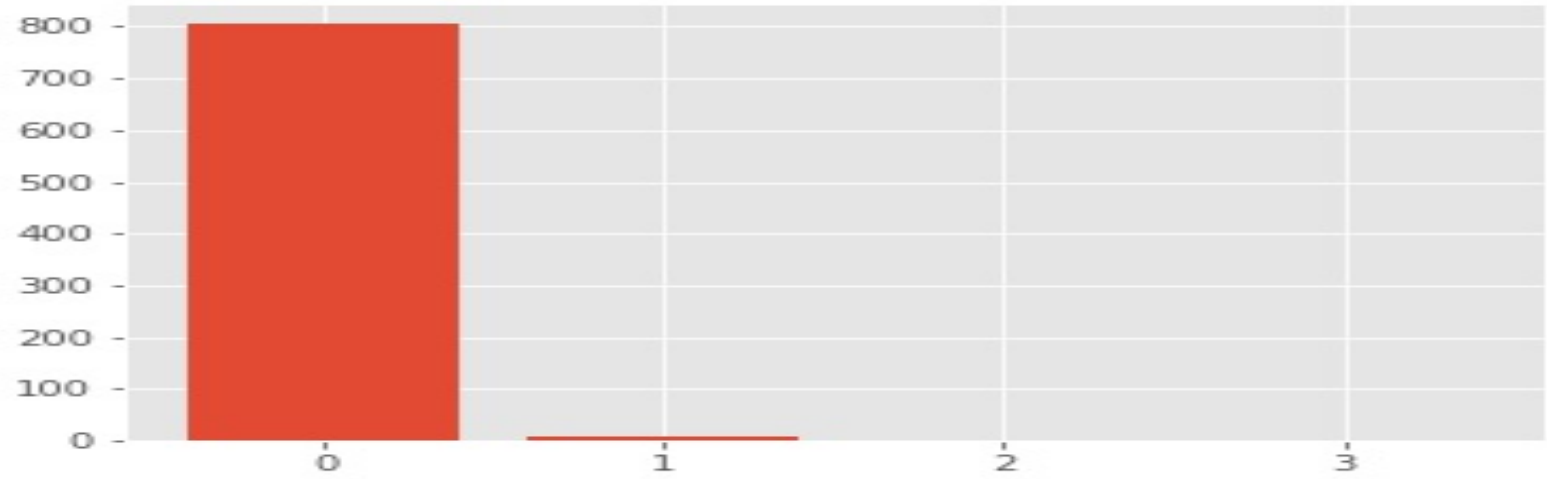
	precision	recall	f1-score	support
Speed limit (20km/h)	1.00	1.00	1.00	19
Speed limit (30km/h)	0.95	0.95	0.95	19
Speed limit (50km/h)	0.95	0.95	0.95	19
Speed limit (60km/h)	1.00	1.00	1.00	19
Speed limit (70km/h)	1.00	0.95	0.97	19
Speed limit (80km/h)	0.86	0.95	0.90	19
End of speed limit (80km/h)	1.00	1.00	1.00	19
Speed limit (100km/h)	1.00	0.89	0.94	19
Speed limit (120km/h)	0.95	0.95	0.95	19
No passing	1.00	0.95	0.97	19
No passing for vehicles over 3.5 metric tons	1.00	1.00	1.00	19
Right-of-way at the next intersection	1.00	1.00	1.00	19
Priority road	1.00	1.00	1.00	19
Yield	1.00	1.00	1.00	19
Stop	1.00	1.00	1.00	19
No vehicles	0.95	1.00	0.97	19
Vehicles over 3.5 metric tons prohibited	0.95	1.00	0.97	19
No entry	1.00	1.00	1.00	19
General caution	1.00	1.00	1.00	19
Dangerous curve to the left	1.00	1.00	1.00	19
Dangerous curve to the right	1.00	1.00	1.00	19
Double curve	1.00	1.00	1.00	19
Bumpy road	1.00	0.89	0.94	19
Slippery road	1.00	1.00	1.00	19
Road narrows on the right	1.00	1.00	1.00	19
Road work	0.95	1.00	0.97	19
Traffic signals	1.00	1.00	1.00	19
Pedestrians	1.00	1.00	1.00	19
Children crossing	1.00	1.00	1.00	19
Bicycles crossing	0.95	1.00	0.97	19
Beware of ice/snow	1.00	1.00	1.00	19
Wild animals crossing	1.00	1.00	1.00	19
End of all speed and passing limits	1.00	1.00	1.00	19
Turn right ahead	1.00	1.00	1.00	19
Turn left ahead	1.00	1.00	1.00	19
Ahead only	1.00	1.00	1.00	19
Go straight or right	1.00	0.95	0.97	19
Go straight or left	1.00	1.00	1.00	19
Keep right	1.00	1.00	1.00	19
Keep left	1.00	1.00	1.00	19
Roundabout mandatory	0.95	1.00	0.97	19
End of no passing	1.00	1.00	1.00	19
End of no passing by vehicles over 3.5 metric tons	1.00	1.00	1.00	19
accuracy			0.99	817
macro avg	0.99	0.99	0.99	817
weighted avg	0.99	0.99	0.99	817

Plot of Accuracy and Loss against the epochs During Training and Validation





Evaluation total statistics (0 - recognized first choice; 1 - second choice; 2 - third choice; 3 - others [unrecognized])



Output for Image:

Output for Video:

Conclusion:

The Traffic sign in an image/video is predicted by the trained models (with an accuracy above 97% each).

- In this project, a traffic sign recognition system, divided into two parts, was presented. The first part is based on classical image processing techniques, for traffic signs extraction out of a video/image, whereas the second part is based on machine learning, more explicitly, convolutional neural networks, for image labeling.
- When put together, the two parts worked well, as the majority of the erroneous data sent by the detector is rejected by the classifier. Voting certainly helps, as more models that behave differently can classify an image and come up with the best statistical result.
- In Night conditions with flash or headlight illumination seem to be the best. The combination of low ambient noise due to lack of ambient light and the high visibility of the sign due to the reflected light makes the detector's job, and, as a result, the classifier's job too, very easy.
- In Shade conditions , the project yield good results, as color is not de-natured and the masks do not cover the signs. The classifier has to reject a moderate amount of noise, most commonly rocks, a very clear blue sky or blue cars, and some patches of grass.
- In Intense light conditions makes the detector skip some signs, their colors being denatured and, as a result, masked by the color mask. In that case, the classifier has no power, being able only to reject the noise it is fed with.
- An improvement that can be brought to classification is a new class, consisting of different kinds of objects that are not traffic signs, so that the models may know how to handle noise better. This would be an alternative to using sigmoid.
- Detection can be done using machine learning as well. Architectures like YOLO can detect and crop images from videos, as well as classify objects in those images, with outstanding speed.

REFERENCES:

- [1]Arturo de la Escalera, Luis Moreno, Miguel Salichs, and J.M. Armingol. Road traffic sign detection and classification. *Industrial Electronics, IEEE Transactions on*, 6:848 – 859, 12 1997.
- [2]A Method for Traffic Sign Recognition with CNN using GPU Alexander Shustanov and Pavel Yakimov
- [3]Miguel Garrido, Miguel-Angel Sotelo, and E. Martm-Gorostiza. Fast traffic sign detection and recognition under changing lighting conditions. pages 811 – 816, 10 2006.
- [4]Pierre Sermanet and Yann Lecun. Traffic sign recognition with multi-scale convolutional networks. pages 2809 – 2813, 09 2011.
- [5]A. Kapoor, N. Nehra and D. Deshwal, "Traffic Signs Recognition Using CNN," *2021 International Conference on Industrial Electronics Research and Applications (ICIERA)*, New Delhi, India, 2021, pp. 1-4, doi: 10.1109/ICIERA53202.2021.9726758.
- [6]Palak and A. L. Sangal, "Traffic Signs Classification using Convolutional Neural Networks: A review," *2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC)*, Jalandhar, India, 2021, pp. 450-455, doi: 10.1109/ICSCCC51823.2021.9478172.
- [7]N. Bhatt, P. Laldas and V. B. Lobo, "A Real-Time Traffic Sign Detection and Recognition System on Hybrid Dataset using CNN," *2022 7th International Conference on Communication and Electronics Systems (ICCES)*, Coimbatore, India, 2022, pp. 1354-1358, doi: 10.1109/ICCES54183.2022.9835954.
- [8] K. Sukhani, R. Shankarmani, J. Shah and K. Shah, "Traffic Sign Board Recognition and Voice Alert System using Convolutional Neural Network," *2021 2nd International Conference for Emerging Technology (INCET)*, Belagavi, India, 2021, pp. 1-5, doi: 10.1109/INCET51464.2021.9456302.
- [9]Z. He, Z. Xiao and Z. Yan, "Traffic Sign Recognition Based on Convolutional Neural Network Model," *2020 Chinese Automation Congress (CAC)*, Shanghai, China, 2020, pp. 155-158, doi: 10.1109/CAC51589.2020.9327830.

THANK YOU