

TRAFFIC SIGN RECOGNITION AND CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORKS

Akshata V. S M.Tech(CSE) Department of Computer Science Information Technology, Jain University
,Jayanagar, Bangalore

Subarna Panda M.Tech(CSE) Department of Computer Science Information Technology, Jain University,
Jayanagar, Bangalore

Abstract: The traffic signs engraved on the roads these days enhance traffic safety by informing the driver of speed limits or any further possible dangers such as deep curvy roads, imminent repair road works or any pedestrian crossings. This paper covers three-stage the real-time Traffic Sign Recognition and classification system, which consists of image segmentation, detecting the traffic sign and a classification phase based on the input image. To extract red regions in the image the colour enhancement technique is used. The detection, classification and recognition are performed using Convolutional Neural Networks (CNN) to identify the content of the traffic signs found.

Introduction: The main purpose of Advance Driver Assistance System (ADAS) in the recent days is to provide the driver with the important information about the traffic signals and warning signs in the road ahead. The actual problem arises when the driver is careless, negligent or completely disobedient of traffic rules and laws. The existing system approach makes sure a safe and comfortable driving experience by developing and giving an accurate road sign detection and recognition system which will forewarn the driver ahead of approaching signs on the road while driving. This system will lower the risk of accidents that can be caused by the driver and will prevent unwanted dangerous situations. For instance, this system will warn the driver in advance if he is about to enter a “Do Not Enter” zone, or helps to avoid the driver from unwanted speeding.

1.1 Problems with TSR

Traffic Sign Recognition (TSR) algorithms face three major problems:

- Due to low resolution the poor image quality, weather conditions that are worse, and over- or under-illumination,
- The signs may sometimes rotate in the wrong direction or may be covered up by some occlusion and or may even get faded.
- The real application such as ADAS comes up with limited memory and processing capacities.

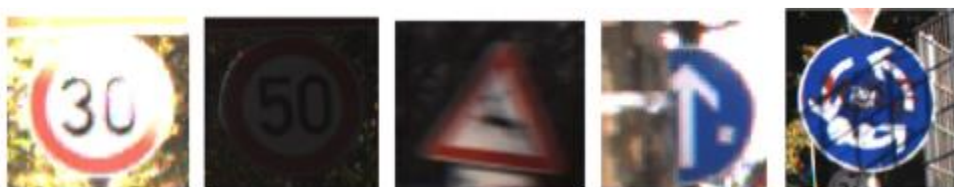


Fig 1.1 Difficulties facing in Traffic Sign Recognition (TSR) task: over-illumination, under-illumination, rotation, occlusion and deterioration of the signs.

1.2 Problem Statement

The performance of the traffic sign recognition systems is improved by using a powerful neural network approach called Convolutional Neural Networks (CNN) which acts as a powerful tool to classify and recognize the traffic signs. In this paper, the machine learning model is designed and trained to classify and recognize the German traffic signs from the dataset provided by the German Traffic Sign Benchmark (GTSB). The coding of this system is done in python language.

1.3 Overview of the Proposed Approach

The main objective is to classify, recognize, and identify the German traffic signs using Convolutional neural networks which are made up of neurons that has learnable weights and biases that helps in giving the high performance in identifying the traffic signs even in its tough vulnerable conditions. This scheme will start by loading the dataset proposed by German Traffic Sign Benchmark. Secondly, it will explore and summarizing each of the data set to visualize them uniquely. Thirdly, to design the training and testing model architecture. And then applying the model to make possible predictions on new images. Finally, to analyse the maximum probabilities of the new images.

1.5 Introduction to Convolutional Neural Networks (CNN)

CNN is one of the neural network models for deep learning, which has three specific characteristics, firstly it takes locally connected neurons, secondly, it calculates shared weight and finally gives the spatial or temporal sub-sampling. Generally, CNN is compromised of two main parts (see Fig.1.2.1). The first contains alternating Convolutional and second is the max-pooling layers. The input of each layer is just the output of its previous layer. As a result, this forms a hierarchical feature extractor that maps the original input images into feature vectors. Then the second part classifies the extracted features vectors, that is, the fully-connected layers, which is a typical feed-forward neural network.

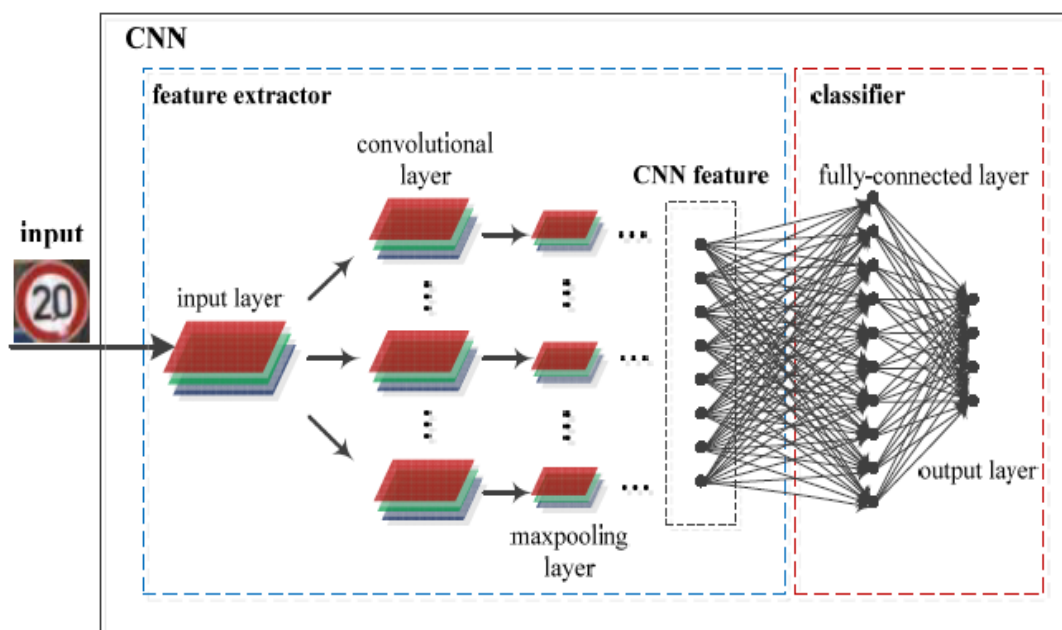


Fig. 1.2 CNN feature extractor architecture.

2. Related Work

Considering work on the basis of various research papers written on the subject are described in this section are as follows:

2.1 Existing Approaches

Most traffic recognition algorithms divide the problem into three stages:

- i. It does a rough segmentation to determine the location of the signs.
- ii. It does a category determination.
- iii. It performs a classification in which it identifies the exact content of the extracted traffic signs using various machine learning techniques.

The segmentation phase is not a mandatory step. However, it is often deployed in approaches using colour images. This section gives a brief overview of some of the techniques used in the three TSR using ELM stages. Some approaches, such as [2,3] reduce the memory and processing requirements by using tracking. The candidates found are tracked over several frames to reinforce the decision made or to reduce the search space in the subsequent frame and therewith accelerate the performance. Some implementations only examine every n th frame, to speed up the overall system.

2.1.1 Determining the Location using segmentation

As traffic signs need to be easily perceivable, they are brightly coloured in red, blue, and yellow. Hence, the detection is often based on a pre-segmentation of the image to reduce the search space and retrieve Regions of Interest (ROIs). Since the direct thresholding of the RGB channels is sensitive to changes in illumination, the relation between the RGB (Red Green Blue) colours is often used. The colour enhancement is used to extract red, blue and yellow blobs. This transform emphasizes the pixels where the given colour channel is dominant over the other two in the RGB colour space. Chromatic and achromatic filters are used to extract the red rims and the white interior of the speed limit and warning traffic signs respectively. The HSI model (Hue Saturation Intensity) is used as it is invariant to illumination changes. Empirically determined fixed thresholds define the range of each HSI channel in which lie the red and blue traffic sign candidates. It is pointed out that HSI is computationally expensive due to its nonlinear formulae.

2.1.2 Category Determination using Shape Detection

Many detection algorithms are based on edge detection, making them more prone to changes in illumination. Edge detection algorithms determine the shapes of traffic signs which exploit the invariance and symmetry of the traffic signs. However, they are often weak in detecting the quality of the pre-processing images, such as extracting the edge or colour segmentation. Circular and triangular signs are detected by the methods called distance transform (DT) and template matching (TM). Similarly, Color Distance Transform is also used, where a DT is computed for every colour channel separately. The advantage of matching DTs over edge images is that the similarity measure is smoother and robust to slight rotations. It is, however, sensitive to affine rotations and occlusions. Support Vector Machines (SVMs) are trained on the Distance to Border (DtB) vectors to classify the shape of an extracted candidate ROI. In [2], the FFT signatures of candidate signs are used to detect relevant shapes. This feature is robust to rotation and scaling, yet not to occlusion and deterioration.

The Hough Transform is also widely used to detect regular shapes such as circles and triangles [4]. The processing time is decreased by the simpler Radial Symmetry Detector method, yet it is limited to circular traffic signs. The Hough Transform results with the use of Confidence-Weighted Mean Shift to eliminate redundant detections. The Hough transform is combined with an iterative process of median filtering and dilation to refine the candidate set. In the detection phase, it uses the approach called gradient orientation information and

classifies the candidate traffic signs by comparing their local edge orientations at arbitrary fixation points with those of the templates. Edge Orientation Histograms are computed over shape-specific sub-regions of the image. The Regions of Interest (ROI) obtained from colour-based segmentation are classified using the Histogram of Oriented Gradients (HOG) feature. To integrate colour information in the HOG descriptor, concatenate the HOG descriptors calculated on each of the colour channels. The advantages of this feature are its scale-invariance, the local contrast normalization, the coarse spatial sampling and the fine weighted orientation binning.

2.2 Proposed System

Convolutional Neural Networks [5] (CNN) use both the combination of supervised and unsupervised learning which is basically a multilayer feed-forward architecture that can learn multiple stages of invariant features. It has various stages wherein each of the stages is composed of different layers such as a non-linear transform layer, a spatial feature pooling layer, and filter bank layer. The spatial resolution of the pooling layer representation is lowered by the spatial pooling layers, therefore making the representation easy to geometric distortions and small shifts, similarly to the “complex cells” in the standard models of the visual cortex. CNN are generally made up of three stages, a classifier composed of one or two additional layers. The upgrade in every single filter and in every filter bank in every layer minimizes gradient-based supervised training procedure by its loss function. In the traditional CNN, the output of the last stage is fed to a classifier. In the present work, the outputs of all the levels are fed to the classifier. And then the classifier is trained to use not just high-level features, that tend to be global, invariant, but also the little precise details, and even it allows using the pooled low-level features, which tend to be more local, less invariant, and more accurately encode local motifs.

Convolutional neural networks (CNN) execute both the feature extraction and the classification [6]. These methods could achieve impressive results but usually on the basis of an extremely huge and complex network, since the fully-connected layers in CNN form a classical neural network classifier, which is trained by gradient descent-based implementations, the generalization ability is limited and sub-optimal.

3. SOFTWARE REQUIREMENT SPECIFICATION

The development and the detailed description of the traffic sign classification and recognition using Convolutional neural networks comes under Software Requirements Specification(SRS) which further describes both functional and non-functional requirements, which may include interactions the users will have with the software that describe a set of use cases.

3.1 System Requirements

3.1.1 Functional Requirements

It includes how Convolutional neural networks works have a classifier of traffic sign which includes the following steps:

i. Determine the data set

Understanding the dataset plays a very important role in functional requirement. According to our problem definition of the paper, downloaded the traffic sign images from the standard dataset provider called German Traffic Sign Benchmark.

ii. Load the data

Once the data set is determined there is a need to load the data set. Here, Jupyter Notebook which allows developing a web application in an open source that uses python language, just by specifying the path where the data set is located the data is loaded from that specified path into the application by the python libraries.

iii. Analyse the data

This part takes care of resizing the images to the appropriate size by specifying the dimensions of the images and rescale, resize the images in order to analyze it programmatically so that the training data has an appropriate size of images.

iv. Data pre-processing

Pre-processing is a necessary step, before the data set is fed into the model, its essential to convert each image of train and test set into an appropriate matrix size and format that converts the class labels into a one-hot encoding vector.

The categorical data is converted into a vector of numbers. The categorical data cannot be implemented directly by the machine learning algorithms. Hence, generating one Boolean column for each category or class. The columns could take only one of these on the value 1 for each sample. This step is crucial one since it specifies the data specific task to the machine learning algorithm and generalizes the model into two parts one for training the data and another one designed to validate the data.

v. Define the Convolutional network

Convert the image matrix into an array and rescale it and feed this as an input to the network. Now, utilise the three Convolutional layers and max-pooling of those layers. (Explained in section 4 and 5)

vi. Model the data

Here, import all the necessary models to train the model which determines the learning parameters and predicts the accuracy of the model.

vii. Compile the model

Usage of Adam optimizer to compile the model once the model is created. This will give the total parameters defined in the model on which the classification takes place.

viii. Train the model

The function is based on which the model is trained. By storing the result of this function, the results with the parameters like its accuracy and performance is analysed.

ix. Model evaluation of the test data set

Our model is evaluated by plotting the graph for accuracy and the loss between the validation and training data.

x. Generating the classification result

Generates the classification result in order to determine the misclassification of data.

3.1.2 Non –Functional Requirements

Non-functional requirements of traffic sign recognition and classification using Convolutional neural networks tells how this system is beneficial for the user.

i. Performance Requirements

In terms of performance, this system uses CNN which has different layers of networks to analyze the data in detail and hence the image acquisition and classification is more clear and accurate, which makes the signal easily recognizable to the user.

ii. Reliability

The system is 99% reliable because this doesn't need any maintenance or preparation for some particular day.

iii. Efficiency

The system is efficient in terms of both memory and time. As it uses CNN, the layering part of CNN, it calculates the minimal memory needed at every layer while processing the image.

iv. Availability

The Camera to capture the images, the database defined within the application and Convolutional neural network classifier is always available any time.

v. Maintainability

The system should be optimized for supportability, or ease of maintenance as far as possible.

4. SYSTEM DESIGN**4.1 System Architecture**

The architecture design part of traffic sign recognition and classification using Convolutional neural networks consists of creating a data set for the application, training the application using CNN, and then classifying and recognizing the traffic signs accurately. The various tools and applications used to design this system are depicted.

4.1.1 Data Set

The images that have used in this paper are downloaded from the standard test data set provided by the German Traffic Sign Detection Benchmark which acts as a supportive mile to

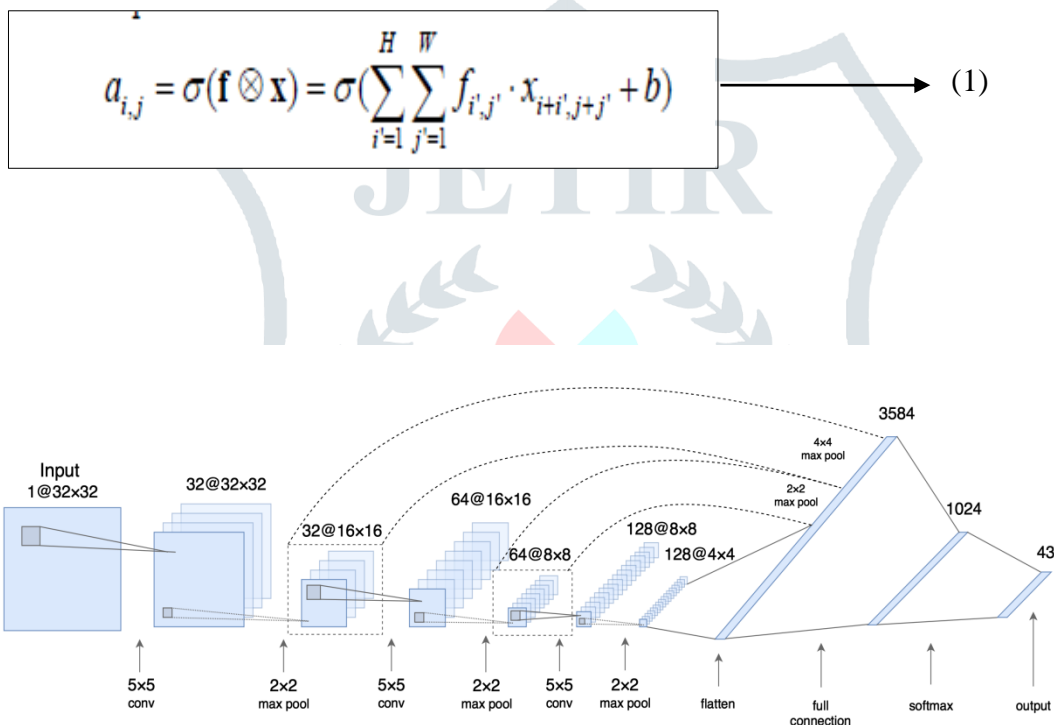
the researchers in the field of computer image processing with respect to research in traffic sign detection.

The training images provided by this dataset are of size 1.6GB which can be downloaded online there are about 900 training images in PPM format (i.e., 1360 x 800 pixels). The images are clear in every perspective with no distortion or bad lighting effects.

4.1.2 Convolutional Neural Networks

Fig 4.1 Model Architecture of CNN

The model used Convolutional neural network has a classifier, wherein each Convolutional layer, each neuron of the layer is connected locally to its inputs of the previous layer, it functions exactly like a 2D convolution with certain filter, then the activation of the layers can be computed as the result of a nonlinear transformation by using the equation (1).



Here \mathbf{f} is a product of $H \times W$ which is a weight matrix of the Convolutional filter, the variable \mathbf{x} indicates that the input neurons are activated. The input neurons are connected with the neuron (i, j) in Convolutional layer, and the corresponding activation function is represented by the $a_{i,j}$. A non-linear activation function is denoted by the symbol $\sigma(\bullet)$ usually called sigmoid or hyperbolic tangent, the bias is denoted by the variable b . The convolution operator is denoted by the symbol \otimes . The second part of CNN is max pooling where the maximum response operation is taken ignoring the convolution to the activations of the neurons where the region is a non-overlapping rectangular region of the previous layer which is given by the equation (2).

$$o_{i,j} = \max(\mathbf{y}) = \max_{1 \leq i' \leq H', 1 \leq j' \leq W'} y_{i+i',j+j'} \quad (2)$$

Here the variable y is the product of $H' \times W'$ which is a neuron patch connected to the neuron (i, j) in the max pooling layer, the maximum response operator is denoted by $\max(\bullet)$, and the output is given by $O_{i,j}$. Here according to the equation the max pooling is non-overlapping and has a stride that equal to the size of the neuron patch, that assures the dimension reductions with the product factor of H' and W' along with each direction. Unlike the other deep learning models, CNN is trained using back-propagation since it reduces the usability of many parameters which were previously used in other neural networks because of the combination of the three architectural characteristics. Therefore, all the parameters in this architecture are jointly optimized through the back-propagation system.

4.2 Proposed Architecture

The proposed architecture mainly uses Convolutional neural networks for execution of both the feature extraction and classification of the data to achieve the accurate impressive results. But before sending the signal images to the CNN, to ensure the illumination invariance which is been subtracted is the average images of the traffic signs and then are forwarded for feature extraction where it preserves the Convolutional and max-pooling layers for the training of the system which classifies and recognizes the images.

4.2.1 Flowchart of the Proposed Architecture

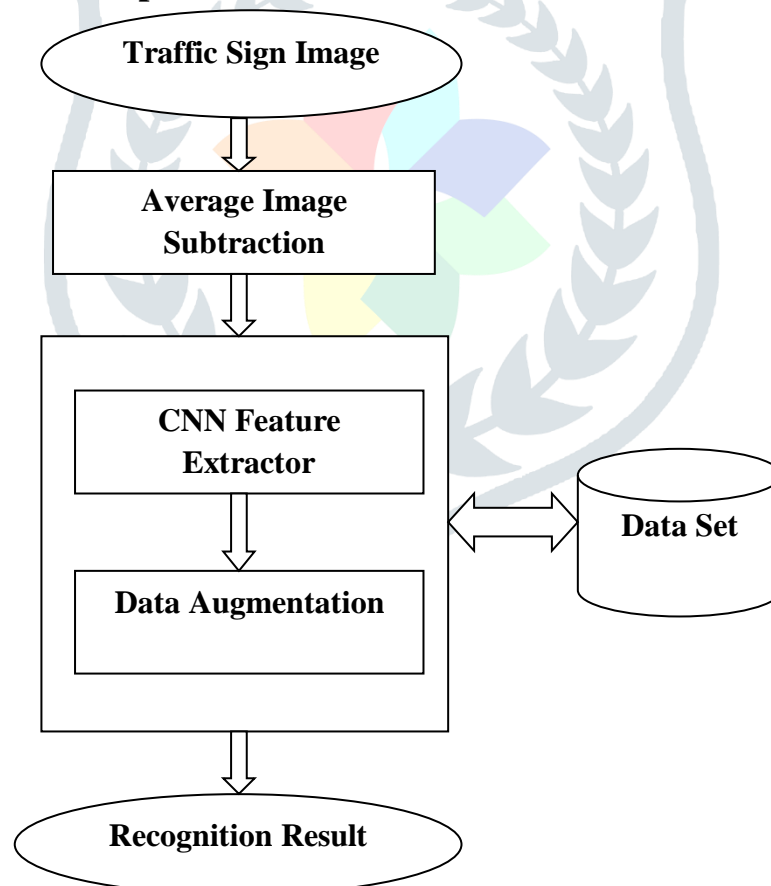


Fig 4.2.1 (a) Flowchart of Traffic Sign Recognition and Classification using CNN

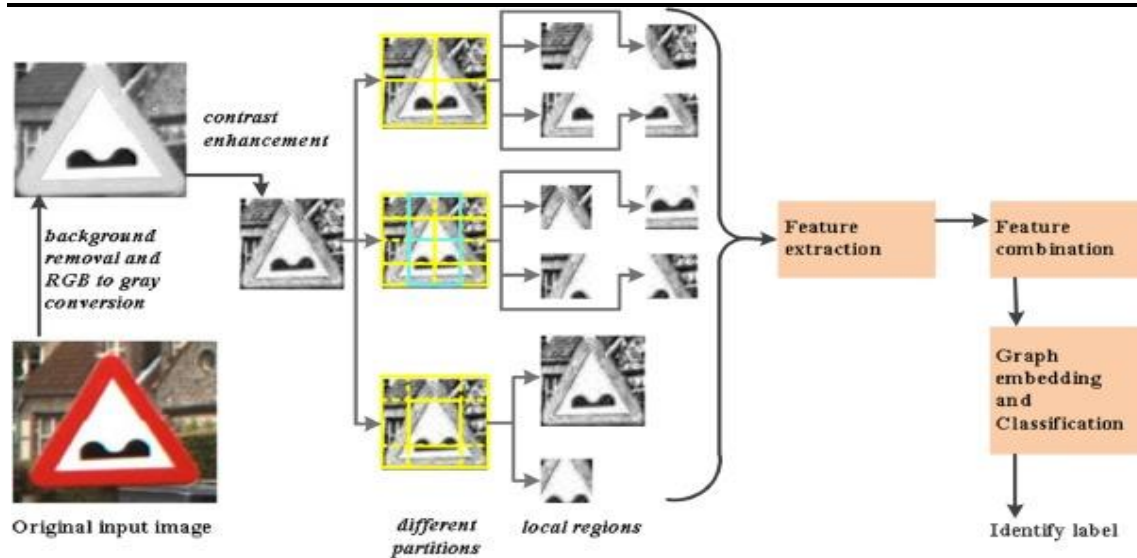


Fig 4.2.1 (b) Flowchart Data Pattern of Traffic Sign Recognition and Classification using CNN

4.2.2 Feature Extraction Using CNN Extractor

In this model, LeNet model is used [7] to train the images because of its simplicity. The models based on deep neural networks with one or two layers which are simpler might have resulted in more hit and trial which would have increased the hyper-parameters list.

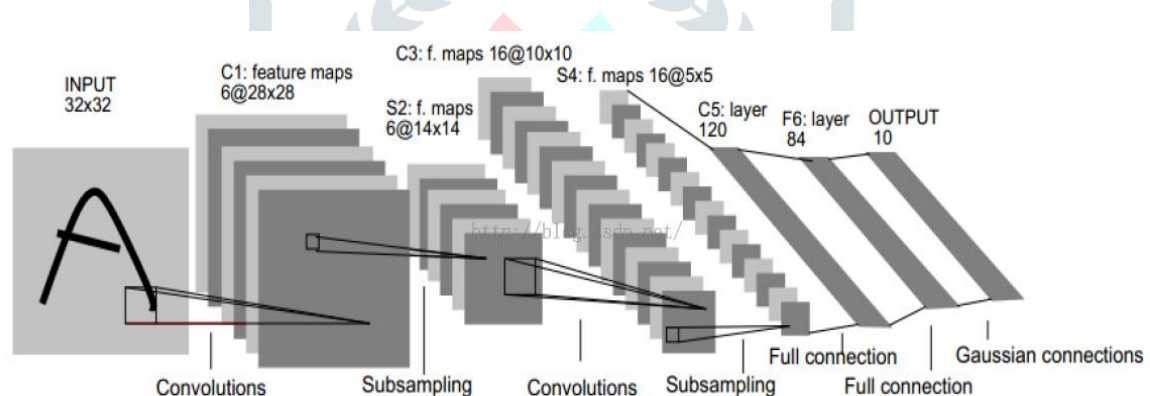


Fig 4.2.2 (a) LeNet Architecture Model

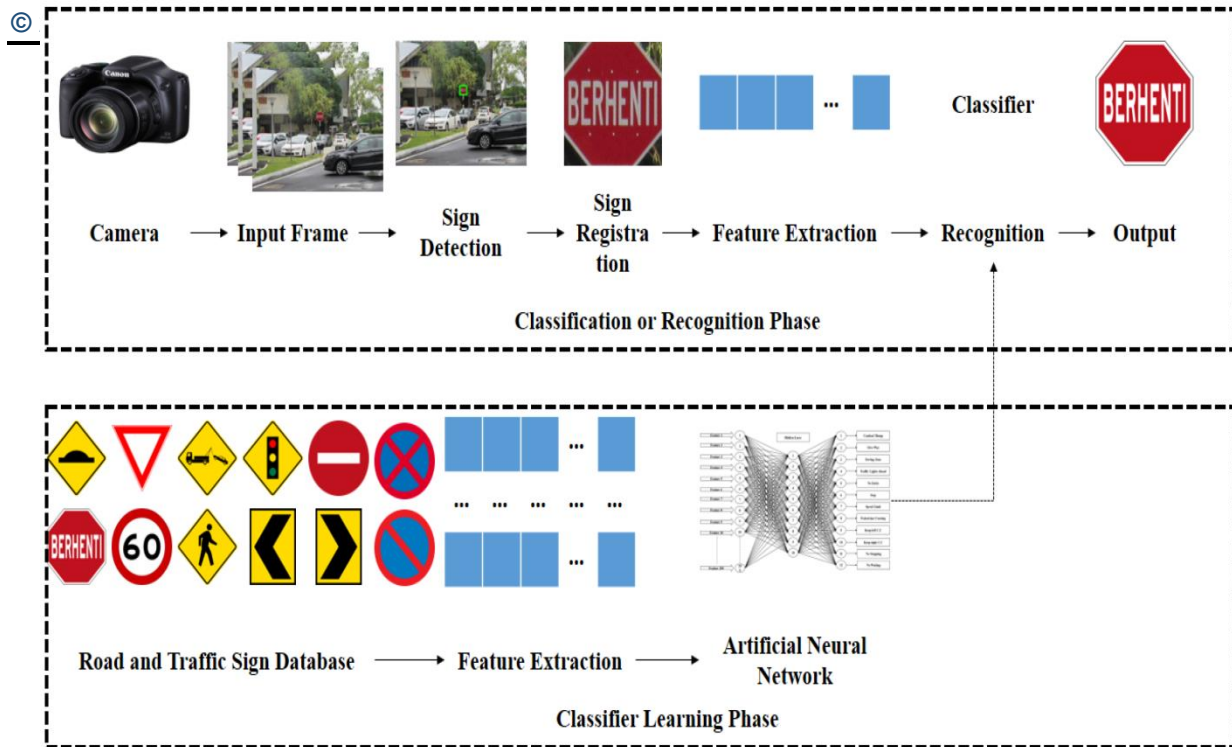


Fig 4.2.2 (b) Symmetrical representation of the working of the model

The current training data is half the size of the original data set. Thereby as a starting point, the obvious choice of the model has been to use the same basic model of [7] with some modification accounting for a number of filters and input types.

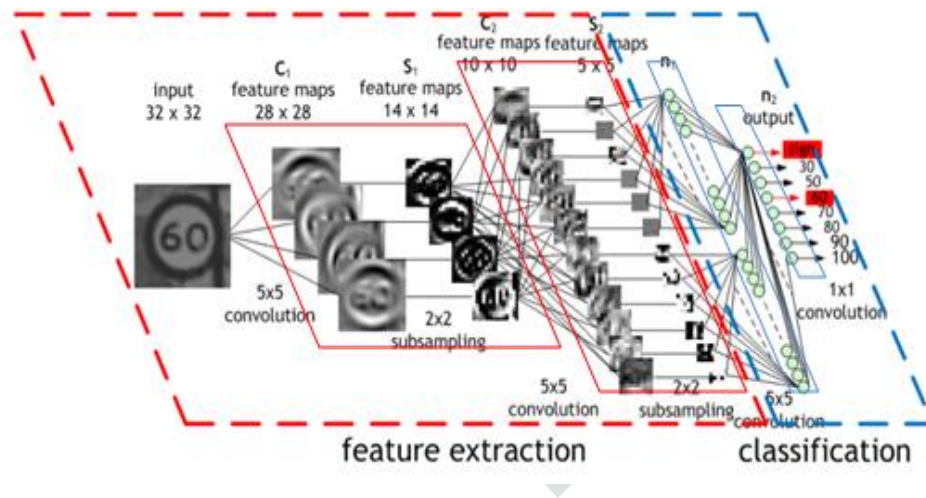


Fig 4.2.2 (c) Traffic Sign Feature Extraction and Classification

Layer 1 Convolutional	
Input	32x32x3 RGB
Kernel	5 x 5 x 6
Output of filtering	28 x28 x 6
Activation	Relu
Pooling	2 x2 with stride of 2
Layer 2 - Convolutional	
Input	14x14 x 6
Kernel	5 x 5 x 16
Output of filtering	10 x10 x 16
Activation	Relu
Pooling	2 x2 with stride of 2
Layer 3- Connected	
Input	5 x 5 x 16 (flattened to 400)
Output	120
Activation	Relu
Layer 3- Connected	
Input	120
Output	84
Activation	Relu
Output Layer	
Output	43
Activation	Relu

Table 4.2.2 CNN Architecture Parameters of LeNet Model

4.2.3 Data Augmentation

Implementation the technique called data augmentation in order to increase the number of images for training data set and it also overcomes the problem of over-fitting. The image augmentation library is used in this work to augment the images.

5. IMPLEMENTATION

The implementation phase of the paper includes the actual practical working of the theoretical knowledge discussed so far on this topic. It covers the topics such as selection of language made to implement the application, the platform used to run the application, and the feasibility of the project, for this paper have made the following choices:

The implementation of this paper involves the following steps:

1. Data Visualization
2. Data Pre-processing
3. Training the CNN model
4. Data Augmentation and training the Model on Augmentation data
5. Introducing and Adding new images to the System
6. Visualization of Layers in CNN and the Final Result

5.1 Data Visualization

The dataset images provided by the GTRB are reduced to the collection of 32 x 32-bit RGB images that have been stored in the paper repository as a pickled dataset. The first cells of the paper in the Jupyter book is defined by a 'un-pickled' datasets and further the three datasets called training, validation and test-sets are obtained. From 43 different classes here are a total of 51389 images.

The breakdown of the images is:

- 34799 number of training samples
- 4410 number of validation samples
- 12630 number of testing samples
- Where each image is RGB with dimensions of (32, 32, 3).
- 43 number of classes.

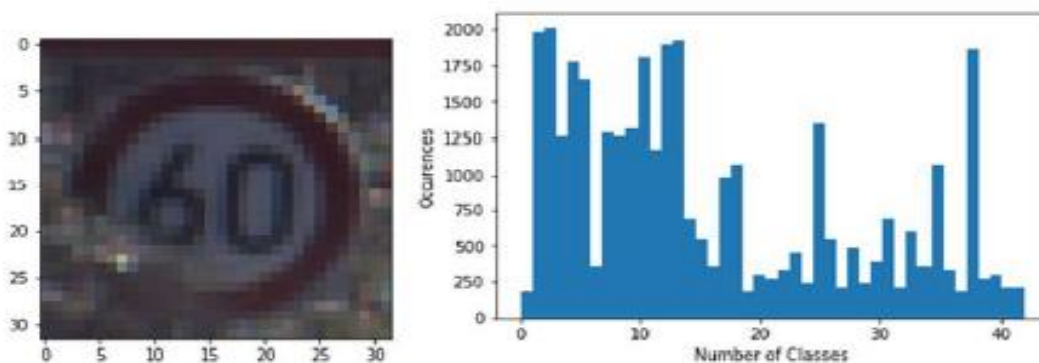


Fig 5.1 (a) A Sample image of the training data (b) The Histogram of the training data

As shown in the figure Fig 5.1(a) that gives a random image from the training data sample, and also gives the histogram of all the image types. The second figure Fig 5.1 (b) gives the graph of how many images per class are available in the training set. The data visualization part of this paper has been done in cell 2 of the Jupyter book.

It is clearly noticeable in the imbalance of a number of images per class in the histogram of the figure. This influences the CNN performance by correctly not detecting the images which have less frequency whereas the images with the high frequency are detected accurately. Hence this directly brings down the overall performance of the CNN leading to low performance.

5.2 Data Pre-processing

The training dataset (TD) contains RGB images, it's been defined a case to convert it to grey-scale, as it makes easier to detect edges images. However, the colour of the traffic sign often gives a strong indication and training the CNN to detect colours will be always advantageous. Therefore, the TD is kept as RGB. However, there is a case to change the dataset into a grey-scale layer to aid training in both colours and edges, which changes the image dimensions to (32, 32, 4). But this approach has not been implemented. Here, by limiting the pixel values of each red (R), green (G), and Blue (B) channels in between the range -1 and 1 the actual pre-processing is done, which is done by subtracting 128 from each pixel and dividing the same by 128. Figure Fig 5.2 shows normalization effects before and after pre-processing.

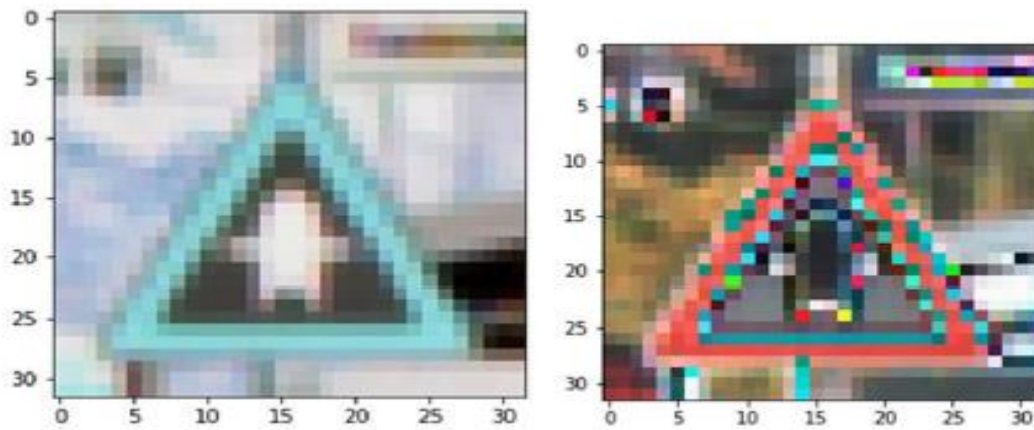


Fig 5.2 Normalization Image before and after Pre-processing

5.3 Training the CNN Model

The Adam optimizer is chosen to reduce the hyper-parameters as learning rate for the Adam optimizer can be left untouched. The other optimizers such as stochastic gradient descent or gradient descent would affect the overall performance with learning rate and hence will require more frequent iterations over the TD.

The number of epochs is set to 30. The whole training time is reduced by setting the epochs. By performing a hit and trial of training dataset, it shows that this number of epochs give a preferable curve of convergence, and as shown in the below later figures it gives the idea about over-fitting. The different batch sizes to be processed are left open to implementing at this moment. The figures Fig 5.3(a), 5.3(b), and 5.3(c) gives the loss and accuracy for validating and training the datasets, here the batch sizes of 64, 128, and 256. Which can be clearly seen in the histogram figure Fig 5.3 (d) where all three batch sizes lead to a validation accuracy level above 95% has been used. Even after training and validating the model, there are signs of over-fitting, which is seen in the figures that the accuracies for all the batch sizes have fluctuations and sharp decreases.

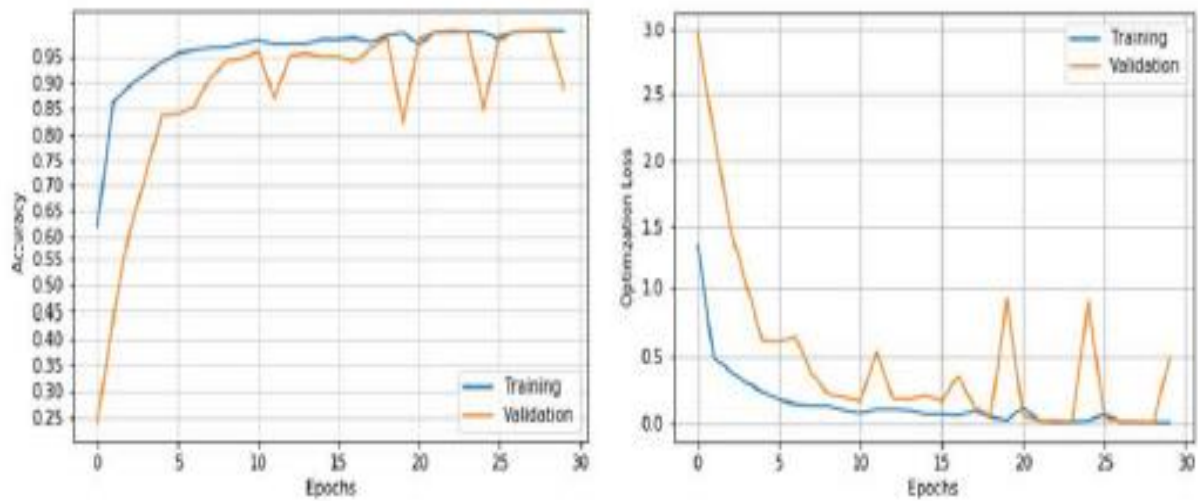


Fig 5.3 (a) Loss plots and Accuracy for batch size 64

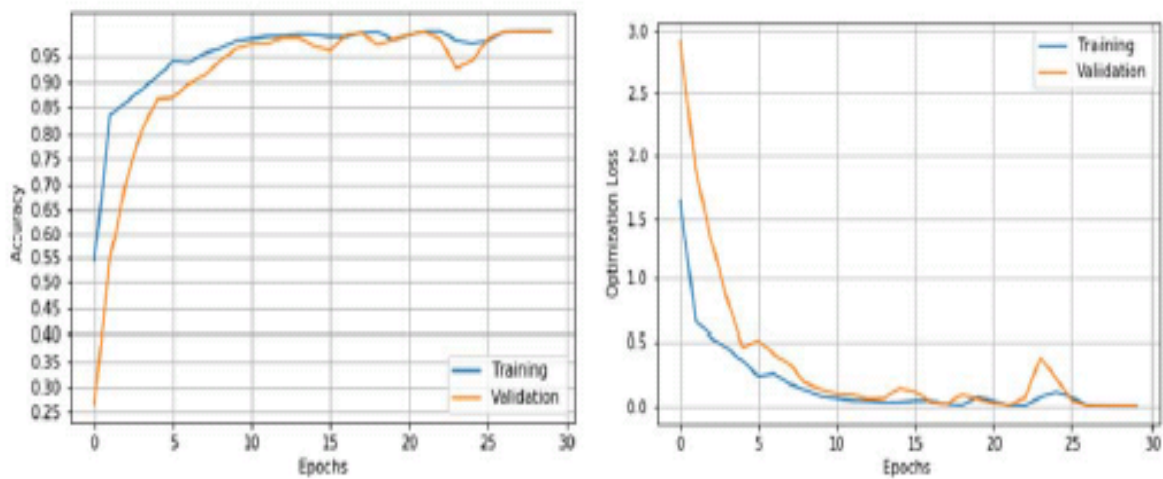


Fig 5.3 (b) Loss plots and Accuracy for batch size 128

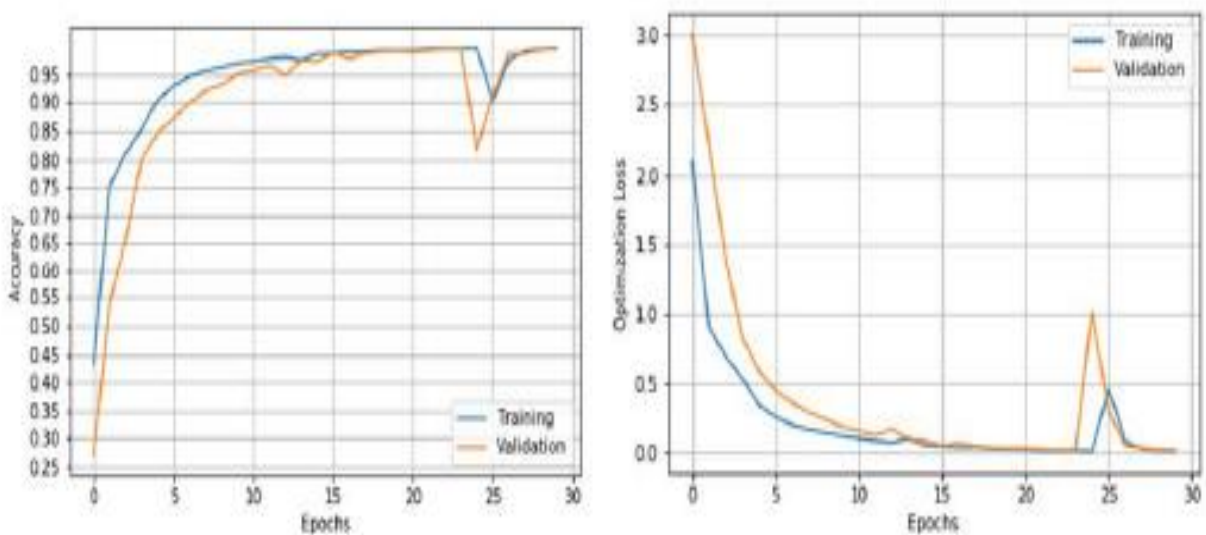


Fig 5.3 (c) Loss plots and Accuracy for batch size 256

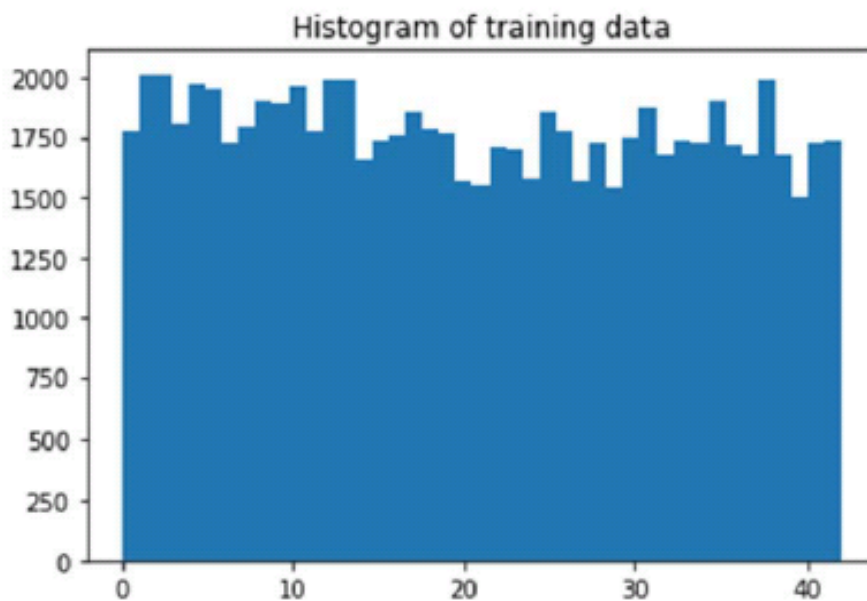


Fig 5.3 (d) Histogram of training data after training the CNN model and Data Augmentation

5.4 Data Augmentation and Training the Model on Augmentation data

In order to avoid over-fitting that occurred while training the CNN model, two methods have been employed. Firstly to increase the number of images for training the data augmentation is employed. Secondly, the drop-off of 50% is introduced on the last two fully-connected layers.

5.4.1 Data Augmentation

Different libraries are present on the internet which can augment images. For this project, the image augmentation library at [8] is used. Any image to be augmented is first subjected to random Gaussian blurring. After which random brightness, random sharpness and normalization of contrast is done on them. The code for this part is in cell 4a of the notebook. The code to obtain the different number of images per class in TD is defined in the cell 4b of the notebook, at first, it determines how many numbers of new images per class should be added. And then the data augmentation is applied to randomly selected images of a class. Overall the number of training data is extended from 34799 to 76500 approximately. The figure is shown above (Fig 5.3 (d)) gives the new histogram for images from all classes.

5.4.2 Training the Augmentation Model

Finally, 96% accuracy is achieved on the test images. The original training data set and the augmented data are combined, and then they are pre-processed, and then they are fed through the model to obtain the 96% of accuracy which is defined in the coding part (Cell 5b) in the notebook.

The final results obtained after 30 epochs are:

- Training Loss is about 14.2 %
- Validation Loss is about 3.6%

- Training Accuracy is about 96.2%
- Validation Accuracy is about 99.2%

5.5 Introducing and Adding New Images to the System

The following five new images have been added to the model from the internet






Traffic Signal	Reason for Choosing the Traffic Sign
	<p>This sign is chosen as it is the most common and frequent sign on the highway. The picture shows a Bad winterish weather. The inside part of the image is slightly blurred which may lead to detection as any other signal which is in circular shape image.</p>
	<p>This sign shows a traffic sign which a camera should be able to capture from the fair distance. The sign shape and number are very clearly visible. This sign is taken as to provide a comparison with the result of the first sign “Speed limit 120 Km/h” because both are circular in shape.</p>
	<p>The “Stop“ sign, shown in the image, this signpost was not on the usual right side of the road but on the left, wherein the right side is covered by a tree. This sign will be a good benchmark to see if the left focusing perspective can be checked by the model.</p>
	<p>The “Yield“ sign is again employed to check for a perspective which is really close and clear under the traffic sign. Which makes the sign to appear un-proportional in size and will make it look broader at the top, and can test the model on this ground.</p>
	<p>The “Wild Animals Crossing” sign offers a tilted perspective, as it is not head on. This image also has a watermark in the middle which can make its correct detection difficult.</p>

Table 5.5 The new images added to the model for testing

The summary of the accuracy and the predicted probabilities of these images are defined inside the Jupyter notebook and the HTML file. The expected output of this experiment is that accuracy on new images is higher, compared to the previous high value of accuracy on the test images. Here 60% accuracy on the 5 new images is observed. The first wrong image that was detected was “the wild animal crossing” sign, which was misclassified and detected as a No Entry sign. The other image which was wrongly recognized was “120km/h limit” sign, which was misinterpreted by the model as a yield sign.

The testing result of this paper makes some sense because in the case that the two wrongly detected signs were not the best quality. The strange thing about those two images has been that the prediction of the correct sign correct sign was not predicted in the top 5 probabilities.

5.6 Visualization of Layers in CNN and Final Result

The visualizing layers of the CNN are defined in code part of Cell 7a and 7b of the notebook. Figure Fig 5.6 (a) shows the input to layer 1 of the CNN while figures Fig 5.6 (b) and Fig 5.6 (c) are the outputs from the convolutions layers having depths of 6 and 16 each.

By observation, it can be seen that the first layer can detect different edges in the figure whereas the batch of filters in the next layer collect more features about the sign, and then it is passed onto the next layers which flattens the information and eventually leads to image detection.

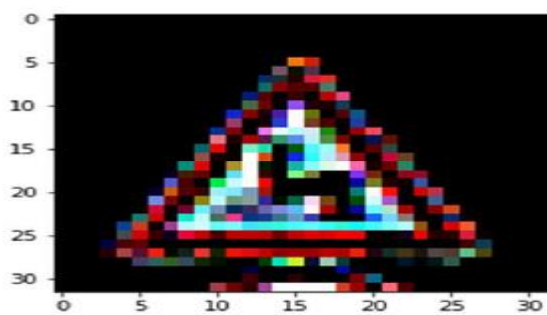


Fig 5.6 (a) An input sample image for the training data

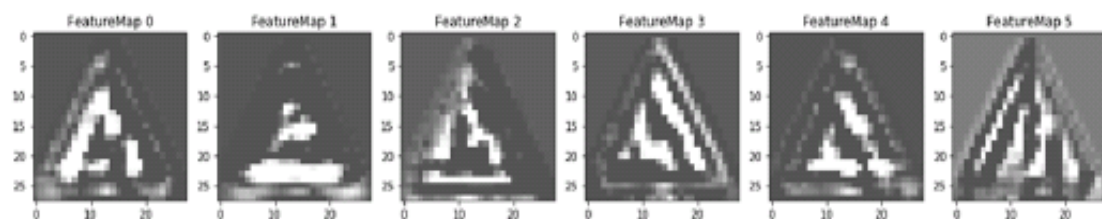


Fig 5.6 (b) Output after the first activation layer

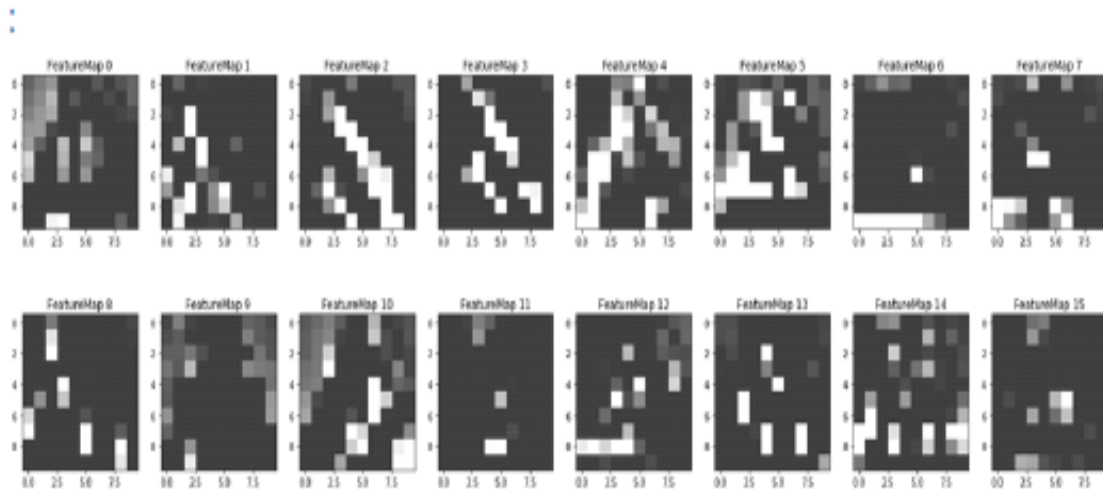


Fig 5.6 (c) Output after the second activation layer

6. CONCLUSION

The experiment results of this paper shows, that the Convolutional neural network is a powerful technique for image classification and recognition. It makes much easier for the researcher to work on it because of its simple architecture, a high-performance classifier for a difficult and much-complicated task such as traffic sign classification and recognition is been obtained. Even though the model is powerful, however, it has some inherent disadvantage such as over-fitting. Hence to overcome this problem, data augmentation is used as a straightforward solution.

REFERENCES

- [1] Fatin Zaklouta and Bogdan Stanculescu, '*Robotics and Autonomous Systems, Real-time traffic recognition in three stages*', Robotics and Autonomous Systems 62(2014) 16-24.
- [2] P.G. Jimenez, S. Lafuente-Arroyo, H. Gomez-Moreno, F. Lopez-Ferreras, S. Maldonado-Bascon, '*Traffic sign shape classification evaluation*', part II, FFT applied to the signature of blobs, in Intelligent Vehicles Symposium, 2005, Proceedings, IEEE, IEEE, 2005, pp. 607–612.
- [3] C. Fang, S. Chen, C. Fuh, '*Road-sign detection and tracking*', IEEE Transactions on Vehicular Technology 52 (5) (2003) 1329–1341.
- [4] W.-J. Kuo, C.-C. Lin, '*Two-stage road sign detection and recognition*', in 2007 IEEE International Conference on Multimedia and Expo, 2007, pp. 1427–1430.
- [5] Pierre Sermanet and Yann LeCun, '*Traffic Sign Recognition with Multi-Scale Convolutional Networks*', Proceedings of International Joint Conference on Neural Networks, San Jose, California, USA, July 31 – August 5, 2011.
- [6] Yujun Zeng, Xin Xu, Yuqiang Fang, Kun Zhao, '*Traffic Sign Recognition Using Extreme Learning Classifier with Deep Convolutional Features*'.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 'Gradient-based learning applied to document recognition' .*Proceedings of the IEEE*, November 1998.

[8] <https://github.com/AkshataVS/imgaug>

PRESENTERS:



ASST. PROF. SUBARNA PANDA
JAIN UNIVERSITY, BENGALURU



ASST. PROF. AKSHATA V S
JAIN UNIVERSIT, BENGALURU

