Convolutional Neural Network Based Traffic Sign Recognition System for Simultaneous Classification of Static and Dynamic Images

Mrunal Padalkar[1, a], *Tejal* Shirsat[1, b], *Snehal* Vekhande[1, c], and *Ekta* Sarda[2, d]

[1]Department of Computer Engineering, Ramrao Adik Institute of Technology, D Y Patil Deemed to be University, Navi Mumbai, India
[2]Asst Prof, Department of Computer Engineering, Ramrao Adik Institute of Technology, D Y Patil Deemed to be University, Navi Mumbai, India

Abstract: Traffic symbols are crucial part of the road infrastructure which are erected at the side of the roads that communicates basic instructions of the road with the help of simple visual graphics which can be understood in no time. As compared to previous decades, traffic congestion is a major issue faced in densely populated cities. Unfortunately, drivers may not notice these traffic signs due to adverse traffic conditions or ignorance which may cause accidents. Therefore, building an intelligent traffic sign recognition model is the need of the time. Besides contributing to the safety and comfort of drivers, traffic symbols recognition has important benefits for autonomous vehicles. In this paper, we have used simple CNN technique to recognize static as well as dynamic traffic symbols on the German Traffic Symbol Recognition Benchmark (GTSRB) dataset which has more than 40 classes of traffic symbols in different orientation and lightning condition. Further, a comparison of performance of CNN on static and dynamic input was done and the efficiency was compared. The experimental results show that the detection rate of the CNN model on static images is 96.15% which significantly higher than that of on dynamic images, which resulted in 95% accurate.

Keywords— CNN, GTSRB, detection, classification.

[a] Corresponding author: mrunalvilas@gmail.com
[b] Corresponding author: 10tejalshirsat@gmail.com
[c] Corresponding author : Snehalvekhande10@gmail.com
[d] Corresponding author : Ekta.sarda@rait.ac.in

1 INTRODUCTION

Road signs or Traffic symbols play extremely pivotal role in facilitating road traffic and increasing road safety through uniform adoption of traffic rules. They also provide special road conditions like curvature, no passing of heavy vehicles, speed limits etc. Unfortunately, drivers may not notice these signs due adverse traffic conditions or ignorance which may cause accidents. Therefore, building an intelligent traffic symbols recognition model is the absolute necessity. The main intention of these models is to provide crucial information for the drivers, in order to minimize their effort and hence result in safe driving. For this, various advance technologies like ML and Artificial Neural Networks are used. Moreover, with advancements and research in the field of Artificial Intelligence various giant companies are developing self-driving cars. For achieving full accuracy and success in this the car needs to be able to interpret the traffic symbols and make proper decisions. Speed Limit, Traffic Signals, No Entry, Direction Indication are some of the many traffic signals we come across usually. Apart from this Traffic sign Recognition has many other real time applications like Self driving cars, Traffic surveillance, Road Network Sustenance, etc. so to build a model that will be able to identify these signs is the motivation of this paper. In this Paper, we have built a Deep Neural Network model that classifies given static and dynamic traffic symbols into different classes. In addition, comparison is made based on efficiency of model on two different types of inputs i.e., Static Images and Dynamic Real-Time Images.

2 LITERATURE SURVEY

Traffic symbol detection and recognition is a system that is created to reduce accidents [3]. The system uses camera to capture the image, which is then processed by an image processor, classifier, and detector before being converted to audio. They used German datasets to classify them using faster R-CNN algorithm. [1] This paper has presented an experimental comparison analysis of eight deep neural network-based traffic sign models. They looked at the most important features of specific detectors, like detection time, precision, accuracy, the amount of floating-point operations in CNN and SPP, as well as the area of the workspace. They developed a real-time system for traffic road sign recognition using SVM algorithm for autonomous vehicles in their work "India Traffic Road Sign Recognition for Intelligent Driver Assistance System using SVM" [4]. In daylight settings, the SBIV function of their system performed effectively in identifying traffic road signs. In the published work [6], the authors Amara Dinesh Kumar R. Karthika Latha Parameswaran has created a model that uses a capsule network to detect pose and spatial variances more effectively than CNN. Even on blurred, rotated, and distorted images the detection performed better. The model presented in the paper [5] enables real-time, high-resolution video processing. This model was developed on the Nvidia K1 CPU, which had CUDA for performance acceleration. A parallel window searching based on GPGPU, which is a real time traffic symbol recognition system for input images with high resolution, was presented in the paper [2]. For stable recognition irrespective of illumination variation, this model includes Byte-MCT. SVM and CNN classifiers were used. In the paper "Traffic Sign Recognition with Convolution Neural Network Based on Max Pooling Positions" [7], a CNN model was created to develop a compact but containing discriminative feature characterization. They have also developed a recognition approach based on MPPs that enhances classification performance and speed. Using MPP's, a unique strategy for improving classification performance and speed has been developed.

3 PROPOSED METHODOLOGY

This paper has proposed a traffic symbol recognition system that will classify both static and Real-Time images using a Deep convolution neural network. An efficient Traffic symbol interpretation system based on CNN is proposed. First, the CNNs a GTSRB ("German Traffic Sign Recognition Benchmark") dataset, which had images that are labelled, rotated and in different lighting conditions. Later, our method performs classification of traffic symbols using CNN; finally (to make it work more efficiently) this model is additionally enhanced to recognize and classify Dynamic Real-Time Images. In comparison to these two kinds of inputs, Static Images when tested by the model provide more efficient outcomes.

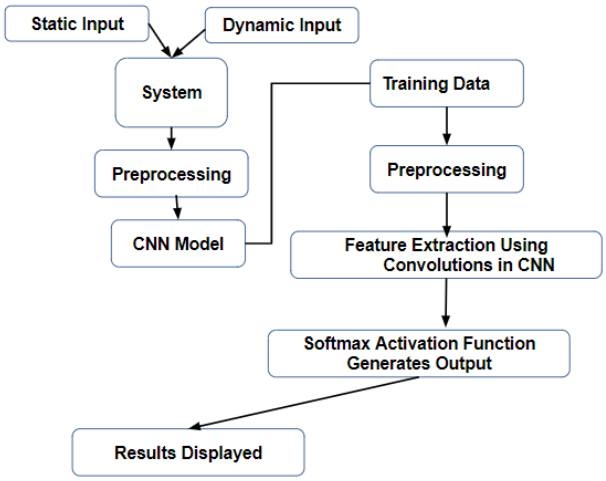

Fig. 1. Proposed Methodology

Fig. 1 Represents Proposed Methodology diagram. In this proposed system, Data Pre-processing is performed for making input images suitable for analysis and operation. Data Pre-processing is different for static and dynamic input. CNN is used for feature extraction, SoftMax layer of CNN is used for classification results.

Paper began by gathering accurate and appropriate datasets for this model. The GTSRB dataset, which included

approximately 50,000 images and 43 classes, was chosen. Various data exploration techniques were then applied to this data, which was then visualized to gain visual insights. In addition, we used CNN to create a classification model. The model is then validated and trained, and a test is run on the developed model. CNN is one of the main technologies used in Neural Networks to perform image identification and image segregation. The classification of images in CNN follows a simple logic: Create hierarchical representations of the data, finding features from a number of images provided and then identifying these features to classify unseen images in specific classes. In the case of Deep Learning CNN models for testing and training of input images, each image is processed through a series of convolutional layers with Kernels, fully connected layers (FC), Pooling, and SoftMax function is applied to these images to classify an object. Deep learning CNN models use probabilistic values ranging from 0 to 1 to express the likelihood of an object pertaining to a particular class. The evaluation of the convolutional layer is expressed in the following equation:

$$a^k = f\left(\sum_{i.j=1}^{3} W^k ij \times X_{ij} + b^k\right) \qquad (1)$$

Where,

$W_{ij}{}^k$ is the weight value filter of convolution,

X_{ij} is the pixel value convolution filter used previously,

b^k is the bias

and the activation function is f(x)

Each step of Proposed methodology is discussed below in brief detail.

3.1. Data Pre-processing

Data Pre-processing is a procedure that is performed on data to format it in the best possible format for subsequent processing. Preprocessing ensures that data from resources is in a common and appropriate format. Because our input data is of two different kinds i.e., Static and Dynamic, it is essential for us to ensure that each image in the dataset has a common dimension; dealing with images of varying sizes is not a feasible choice. As a result, we resized the entire input to a fixed size of 30 x 30. Furthermore, we appended the appropriate labels to the image and converted the image content into an array for submitting to the model. Hence, the data dimension is (39208,30,30,3), and there are 39208 images with dimensions of 30 x 30 and they are colored with RGB (3) channels. To gain a better understanding of the dataset, we used exploratory data analysis techniques to visually summarize it. Because the dataset contains many attributes, the set of correlation values between pairs of its attributes is represented in the correlation matrix, and a heat map of this correlation matrix was plotted later to visually understand the attribute linearity. Data preprocessing simplified subsequent processing and gave us an understanding that the data was enough for applying any deep learning algorithm.

3.1.1. The human brain does not interpret an image by pixel, but rather processes visual signals through a structure of multiple layers. As neural networks represent these structures well, we opted for neural networks in this model. An architecture with insufficient depth may necessitate the addition of many more computational elements. Therefore, CNN algorithm with increased depth and layer(pooling) for transition of invariant features is built. Because the fully connected layers include the majority of the network's parameters, they are frequently prone to overfitting. To prevent this dropout layer is added. Finally, as there are 43 different classes to categorize, the model is built using cross entropy measures.
3.1.2. After successfully building a hierarchical CNN model, our model was trained on training set, validation set, multiple epoch values, batches sizes, and 2 activation functions for the classifier. In case of Static images, the batch size of 32 outperformed other batch sizes with 10 epochs and on the other hand batch size of 50 with 10 epochs worked best for Real-Time images; the accuracy was stable. Training set gave an accuracy of 96.15 percent on Static input and 95% on Dynamic input. An analysis of the accuracy and the loss was conducted.
3.1.3. Finally, now that the model has been trained. A GUI model using Tkinter was created, which provides effective interface to classify static and Real-Time Traffic sign images. It takes as input static images and classifies the traffic sign present on them and also allows user to start Camera equipment to capture Real-Time scene and recognize Traffic Symbol from it.
3.1.4. In case of Dynamic Images, The live stream Detects the Traffic Symbol and Displays the Meaning and probability of Classification of that Traffic Symbol on the screen itself.

3.2 System Diagram for CNN Model

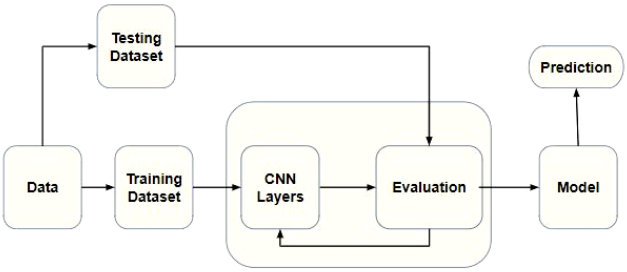

Fig. 2. System Diagram of CNN Model

Fig. 2 depicts an overview of the simple CNN model that was used. The initial dataset is divided into train, test and validate folders**,** and then images in the train folder are each passed through the CNN model layers, and the model with the help of convolution identifies the features and develops a feature map. It extracts these features, and finds interrelationships between them, before passing this feature extracted and reduced dimensional data through the activation function for final decision making. Having many convolution layers in the network allows the network to Develop representations of objects in the images from simple features to more complicated features and up to sensitivity to distinct categories of objects. When the data from the test folder is tested against the trained model, the activation function evaluates the

probability of each test image belonging to one of the many categories of traffic symbols, and the one with the highest probability is displayed in the frontend screen and a prediction is made.

3.3 System Diagram for Classification of Static Images

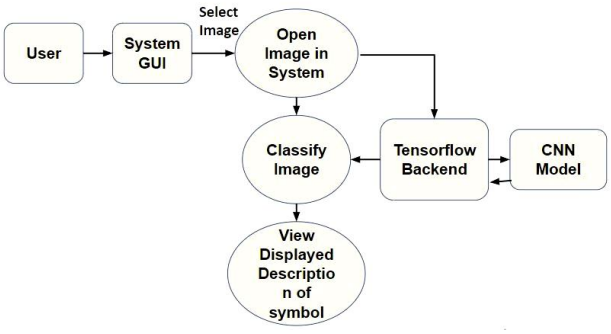

Fig. 3. Proposed System Diagram For Classification of Static Images

The Fig. 3 depicts our system's diagram for classification of static images. To use the system, the user must first start up the TensorFlow backend. Once the backend is activated from the command line, the GUI loads and the user can see an upload button and a button for Real-Time classification, upon pressing upload image button the user can input image through file system in his/her system and can upload any image in .png or .jpg format upon loading the input image in the system classify button will appear upon pressing which the user activates the CNN Model through TensorFlow backend and the image is resized to 30 x 30 image and then fed to the model. After processing the image, the model generates a list of decimal values indicating the probability that the image belongs to one of the 40+ classes. The SoftMax function then chooses the class with the highest probability, and thus a static traffic symbol image is successfully interpreted by our simple CNN Based system in no time.

3.4 System Diagram for Classification of Dynamic Images

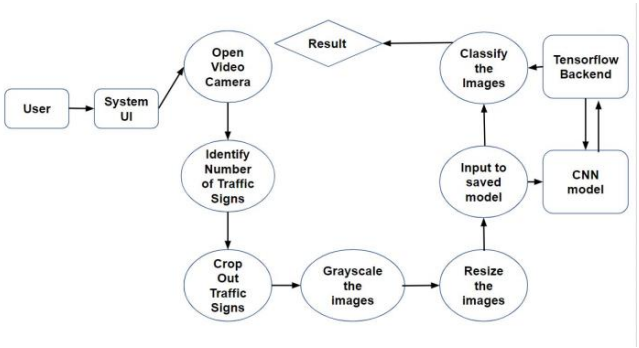

Fig. 4. Proposed System Diagram for Classification of Dynamic Images

Fig. 4 represents details For Dynamic or Real-Time Recognition, Classify Video button provided on System User Interface must be hit for dynamic or real-time recognition, which Turns the camera Equipment of the System the user is using, allowing a live scene to be taken. Red Rectangles indicate which region is identified as traffic sign. The class the Probability value of that particular image is displayed on the screen. The camera will continue to collect the live footage until the user tells it to stop. What happens in the background for real-time traffic sign classification is, when user starts the live image capturing, First Identification of Number of Traffic Signs present on the screen is performed, which is followed by cropping out the required region containing Traffic sign is done, which is then followed by preprocessing of that required region and feeding that image frame to the model is performed, this entire process results in detection of the traffic symbol from live stream and displaying the meaning and probability of classification of that traffic symbol on the screen.

4 RESULTS AND SIMULATIONS

4.1 Stages of Implementation

Stage 1:

Gathering the Input data to work on is done in this stage 1.

Stage 2:

In stage 2, for Static Input we use PIL library to open image content into array. For Dynamic Input OpenCV library is used. Now that we have images in the form of a list i.e., data and labels, we convert it to array for further processing. The system then summarizes and analyze the data using Data Analysis techniques. The images are in colored form. The variable explorer shows that we have 39208 (pictures) with a height of 30 px, a width of 30 px, and in RGB color format.

	Width	Height	Roi.X1	Roi.Y1	Roi.X2	Roi.Y2	ClassId	Path
0	27	26	5	5	22	20	20	Train/20/00020_00000_00000.png
1	28	27	5	6	23	22	20	Train/20/00020_00000_00001.png
2	29	26	6	5	24	21	20	Train/20/00020_00000_00002.png
3	28	27	5	6	23	22	20	Train/20/00020_00000_00003.png
4	28	26	5	5	23	21	20	Train/20/00020_00000_00004.png

Fig. 5. Head Function on GTSRB Dataset

	Width	Height	Roi.X1	Roi.Y1	Roi.X2	Roi.Y2	ClassId	Path
39204	52	56	5	6	47	51	42	Train/42/00042_00007_00025.png
39205	56	58	5	5	51	53	42	Train/42/00042_00007_00026.png
39206	58	62	5	6	53	57	42	Train/42/00042_00007_00027.png
39207	63	69	5	7	58	63	42	Train/42/00042_00007_00028.png
39208	68	69	7	6	62	63	42	Train/42/00042_00007_00029.png

Fig. 6. Tail Function on GTSRB Dataset

Fig. 5 and 6 show the details of the dataset by revealing the starting and ending 5 rows, respectively, as well as its characteristics. The dataset has 39208 rows and 8 columns.

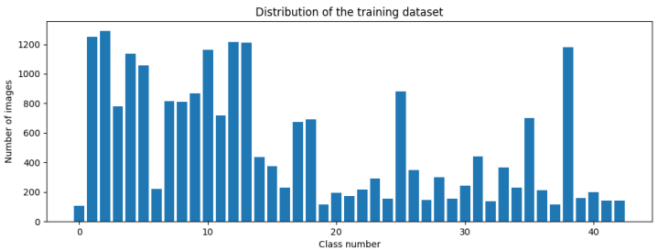

Fig. 7. Bar graph representing number of images per class

Fig.7 Represent the Distribution of Images among 43 classes. It can be seen that the distribution is Un-even and hence we might get good classification for one class and bad classification of other. It is Observed that the class with minimum 500 images had had great classification accuracy. Classes with less than 500 images receive poor classification accuracy.

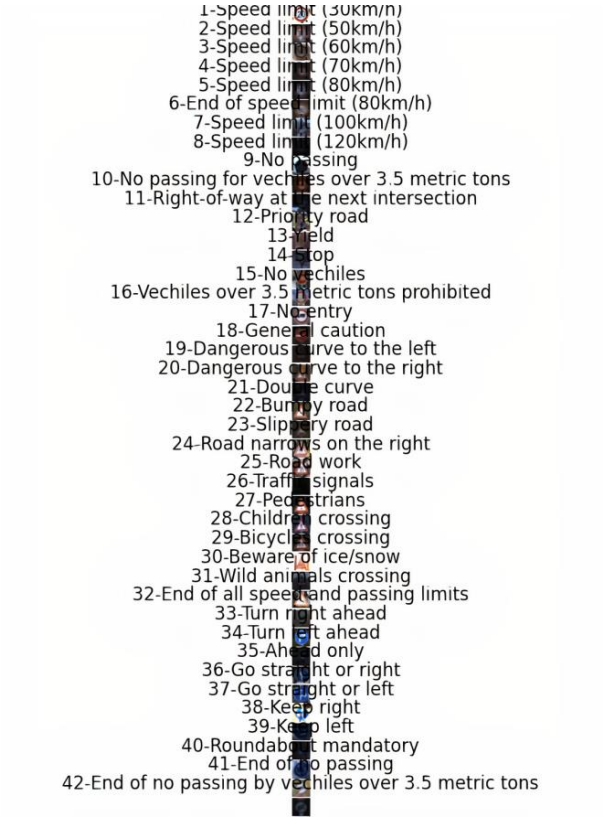

Fig. 8. Classes with their corresponding images

Fig. 8, gives a brief Idea about all the Images Present in the Dataset with Their corresponding classes marked by Class-ID. Division of data into train, test and validate set is performed. Neural networks expect the labels of classes in a dataset to be organized in a one-hot encoded manner. Conversion of labels into one hot encoding is also executed.

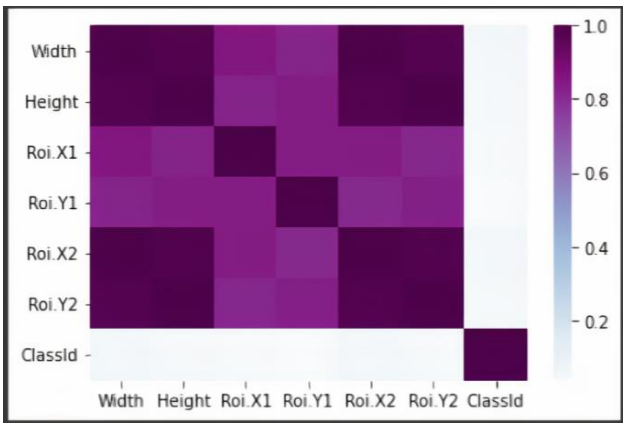

Fig. 9. Relationship Analysis using Heatmap

Summarization of the data for more advanced analysis is provided by correlation matrix. Fig. 9 depicts a heatmap-based visual representation of the correlation matrix. The data in a Two-Dimensional form is represented using heatmap. Heatmap is serves as a way of representing the data values in colored graph.

Stage 3:

Data Preprocessing is performed at stage 3. First Images are converted to Grayscale to reduce the dimensionality. As we all know grayscale images have only one channel this helps the model to focus on important parameters. Gray scaling is followed by Equalization phase. Equalizing is important to ensure that all images have standard Lighting. Normalization is last step in Preprocessing, here Values are normalized by dividing each image by 255, so resulting values are between 0 and 1 instead of 0 to 255.

Fig. 10. Augmented Images

Stage 4:

Augmentation of Image is performed in this stage 4, to ensure that the images are generic and not plain. Images are Shifted left and right, zoomed in and rotated. Fig.10 Represents Augmented Traffic Sign Images.

Stage 5:

In Stage 5, CNN Model is Built from Various Layers. Keras was used to create and train the neural networks to classify the

images. This model will be of Sequential Type, meaning output of one layer will be fed as input to next layer. Convolutional networks for classification are constructed for two purposes: for image processing and for readout from a sequence of convolutional layers and fully connected layers. Few Convolution layers, few pooling layers and dropout layers and stacked and at the end a Dense layer is used as an Output layer with 43 possible outputs.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 60)        1560
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 24, 24, 60)        90060
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 12, 12, 60)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 10, 10, 30)        16230
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 30)          8130
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 4, 4, 30)          0
_________________________________________________________________
dropout (Dropout)            (None, 4, 4, 30)          0
_________________________________________________________________
flatten (Flatten)            (None, 480)               0
_________________________________________________________________
dense (Dense)                (None, 500)               240500
_________________________________________________________________
dropout_1 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_1 (Dense)              (None, 43)                21543
=================================================================
Total params: 378,023
Trainable params: 378,023
Non-trainable params: 0
```

Fig. 11. Summary of CNN Model

Fig. 11 represents the summary of the CNN model that was created. A deep convolutional neural network is a network that has multiple layers. Each layer in a deep network receives its input from the layer before it, with the very first layer receiving its input from the images used as training or test data. We used a stack of 11 neural networks for extracting features and for making model learn their inter relationships.

Stage 6:

The model is trained and validated with multiple batch sizes in stage 6, training set and validation set. The model performs better with 32 batch size for static images and 50 for dynamic images than with other batch sizes, and the accuracy is stable when the epoch is set to 10.

```
Epoch 1/10
256/256 [==============================] - 92s 357ms/step - loss: 3.4273 - accuracy: 0.0882 - val_loss: 1.5344 - val_accuracy: 0.5413
Epoch 2/10
256/256 [==============================] - 90s 353ms/step - loss: 2.0026 - accuracy: 0.4034 - val_loss: 0.8338 - val_accuracy: 0.7707
Epoch 3/10
256/256 [==============================] - 95s 370ms/step - loss: 1.3885 - accuracy: 0.5787 - val_loss: 0.4779 - val_accuracy: 0.8860
Epoch 4/10
256/256 [==============================] - 99s 388ms/step - loss: 1.0433 - accuracy: 0.6745 - val_loss: 0.2749 - val_accuracy: 0.9280
Epoch 5/10
256/256 [==============================] - 131s 511ms/step - loss: 0.8451 - accuracy: 0.7395 - val_loss: 0.2246 - val_accuracy: 0.9449
Epoch 6/10
256/256 [==============================] - 111s 434ms/step - loss: 0.6909 - accuracy: 0.7816 - val_loss: 0.1957 - val_accuracy: 0.9551
Epoch 7/10
256/256 [==============================] - 122s 478ms/step - loss: 0.6006 - accuracy: 0.8094 - val_loss: 0.1461 - val_accuracy: 0.9565
```

Fig. 12. Each Iteration Details

Fig.12 Represents the Accuracy and loss of the model at each Iteration. It can be seen that accuracy is increasing with each epoch.

Stage 7:

Now that the data is trained, In stage 7 we tested our model. We begin by resizing the height and width to 30x30 and creating a NumPy array of the test images. The accuracy score is imported. Lastly, we save the model. Data visualization of accuracy and loss is done. During learning process, the network observes change in weights, and as weights change, the network becomes more adjustable to the features of the images that differentiate between classes. This means that the used loss function becomes smaller and smaller. Observing the change in the loss with learning can be useful to see whether learning is progressing as expected. For progressive learning loss function must go down.

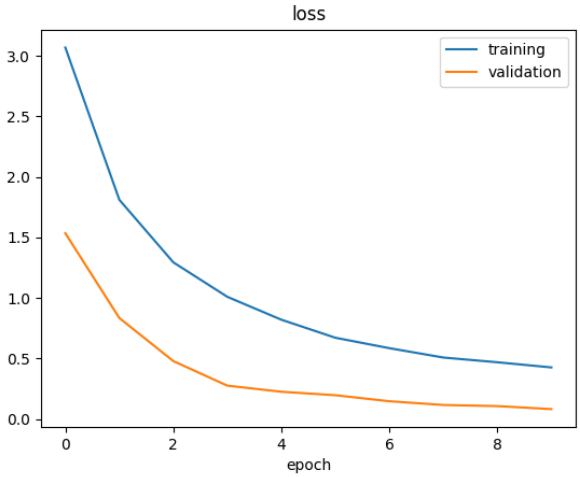

Fig. 13. Plot of training set loss and validation set loss

Fig. 13 is a curve showing the categorical cross-entropy loss in a network that is learning to classify different types of traffic signs, measured on the training set and validation set. You can see that there are rapid decreases in loss in the first few epochs of training, after which learning slows down. This is a sign that learning is going rather well. For validation loss learning is progressing in similar fashion and it never flattens out, this means when validated against a separate set of data, loss becomes better and smaller.

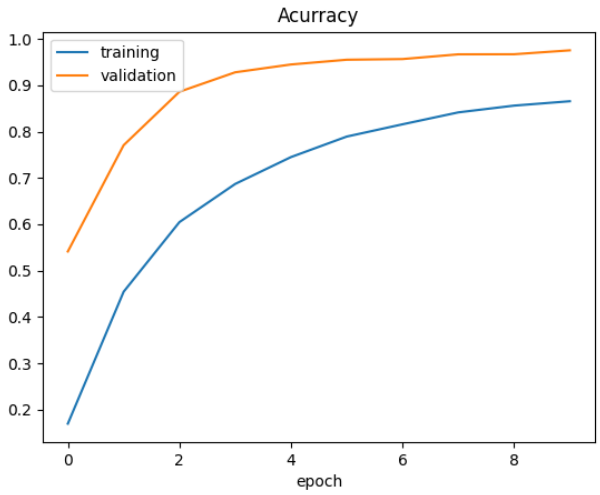

Fig. 14. Plot of training set accuracy and validation set accuracy

Given Figure 14 is a curve showing the accuracy in a network that is learning to classify different types of traffic signs, measured on the training set and validation set.

Stage 8:

Stage 8 represents the GUI application for the model. This model is built using Tkinter module.

Fig. 15.1 GUI Of proposed System

The GUI of the deep learning model is shown in figure 15 and its sub-figures. As shown in Fig. 15.1, The system has two buttons: upload an image and classify video. The Upload an Image button will take us to the test folder, where we can select any random image from a pool of over 40 classes. After uploading the image, when you click the classify image button, the image is correctly classified. For example, if we have uploaded a general caution road sign in the GUI the system will correctly predict its label. If user opted for Classification of Video option, Camera Equipment starts to capture the live scene and Traffic Symbol is detected and classified in Real-Time Environment. OpenCV Library is Used to set the Camera Parameters for inputting live feed. The accuracy fluctuates in the training set, but it is stable in the validation set.

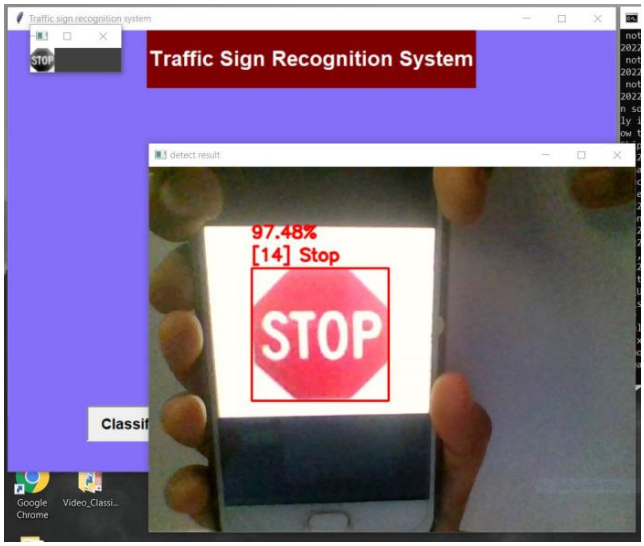

Fig. 15.2 Classification Result of Dynamic Image

Fig 15.2 Displays the results of Dynamic Image Classification, the traffic sign is captured live from camera feed and the class of that particular traffic sign is shown with its probability. Hence, we can conclude that proposed system is able to detect traffic sign in a live video and the traffic sign is highlighted by red rectangle. The class of that image is displayed on top left corner along with its probability percentage are displayed.

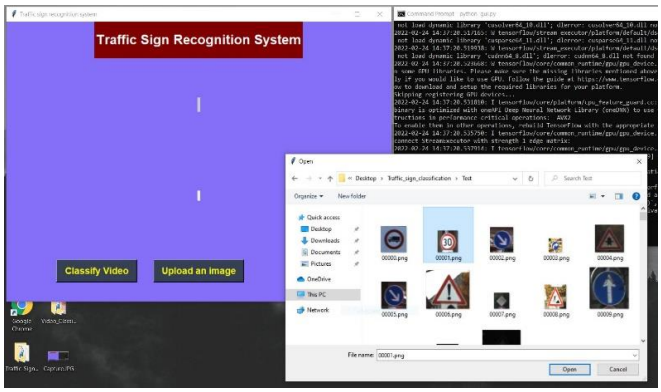

Fig. 15.3 Input static image from proposed system

Fig 15.3 Displays how system allows for selection of static images from system. User can upload one image at a time. The image formats supported by system are .png and .jpg

Fig. 15.4 Classification Result of Static Image

Fig 15.4 shows result of static image recognition. The chosen traffic sign is displayed on the screen along with the class it belongs to.

4.2 Results:

Table 1. Classification Accuracy of CNN Model on static and Dynamic input

Input Type	Accuracy
Static Image	96.15%
Real-Time Dynamic Images	95%

As shown in Table 1, Classification accuracy for static input is 96.15% and, Classification accuracy for Dynamic input is 95%. Accuracy for static input is more than that of dynamic image input.

5 CONCLUSION

Thus, This Paper has been successful in Development of an efficient Deep CNN Based traffic sign recognition system for classification of both Static and Dynamic Input. The color information present in the image, as well as the dimensional property of the road symbols, are used to classify the identified traffic symbols. According to the implementation, the system can grab a high recognition rate of 96.15 percent for static images and 95% for Real-Time Images. The system delivers accurate results in different weather, lighting, and daylight conditions This System is relevant for static as well as dynamic images, but accuracy of dynamic image classification can be improved, which becomes the Future scope for this paper.

6 REFERENCES

1. Dewi, C., Chen, R. C., & Tai, S. K. (2020). Evaluation of robust spatial pyramid pooling based

on convolutional neural network for traffic sign recognition system. Electronics, 9(6), 889.

2. Lim, K., Hong, Y., Choi, Y., & Byun, H. (2017). Real-time traffic sign recognition based on a

general purpose GPU and deep-learning. PLoS one, 12(3), e0173317.

3. Anju, C. P. Traffic Sign Detection and Recognition.

4. Sathya, R., Thiruvenkatasuresh, M. P., Arieth, R. M., Revathy, G., & Babu, M. D. V. (2020).

INDIAN TRAFFIC ROAD SIGN RECOGNITION FOR INTELLIGENT DRIVER ASSISTANCE SYSTEM USING SVM. Journal of Critical Reviews, 7(9), 3177-3185.

5. Shustanov, A., & Yakimov, P. (2017). CNN design for real-time traffic sign recognition. Procedia engineering, 201, 718-725.

6. Amara Dinesh Kumar R.Karthika Latha Parameswaran (2018) .Novel Deep Learning Model for Traffic Sign Detection Using Capsule Networks.

7. Qian, R., Yue, Y., Coenen, F., & Zhang, B. (2016, August). Traffic sign recognition with convolutional neural network based on max pooling positions. In 2016 12th International conference on natural computation, fuzzy systems and knowledge discovery (ICNC-FSKD) (pp.578-582). IEEE.

8. Sheikh, M. A. A., Kole, A., & Maity, T. (2016, October). Traffic sign detection and classification using colour feature and neural network. In 2016 International Conference on Intelligent Control Power and Instrumentation (ICICPI) (pp. 307-311). IEEE.

9. Kale, Amol Jayant, and R.C. Mahajan. "A road sign detection and the recognition for Driver Assistance Systems", 2015 International Conference on Energy Systems and Applications, 2015.

10. Fang, C.Y.. "An automatic road sign recognition system based on a computational model of human recognition processing", Computer Vision and Image Understanding, 200411.

11. Abderrahmane Adoui El Ouadrhiri, Jaroslav Burian, Said Jai Andaloussi, Rachida El Morabet, Ouail Ouchetto, Abderrahim Sekkaki. "Chapter 34 Fast-Tracking Application for Traffic Signs Recognition", Springer Science and Business Media LLC, 2018.

12. Jharna Majumdar, N R Giridhar, P E Gagan. "Autonomous Mobile Robot Navigation on Identifying Road Signs using ANN", 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)