JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Traffic Signs Recognition

Mentor: Prof. Sharada H N, Prof. Sandhya S V

[1]Bharatesh Nagaraj Labhagond, [2]Sharanabasava, [3]Anirudh Gudisagar, [4]Rajesh

[1] B.E Student, [2] B.E Student, [3] B.E Student, [4] B.E Student
Department of Computer Science and Engineering,
SDM College of Engineering and Technology, Dharwad, Karnataka, India.

Abstract: Traffic signs are mandatory features of road traffic regulations worldwide. Automatic detection and recognition of traffic signs by vehicles may increase the safety level of drivers and passengers. For this reason, Real Time-Traffic Sign Recognition system is one of the essential components for smart transportation systems and high-tech vehicles. Recently, very good performances have been achieved in public datasets, especially with advanced Computer Vision (CV) approaches like Deep Learning or more precisely CNN (neural networks) due to high recognition rate and fast execution.
CNN has largely influenced all the computer visionary tasks. So, in this project we propose a deep network traffic sign recognition/classification model with the help of python as the base language and followed by different python libraries for training the CNN model. This model will consist of different CNN layers which will precisely classify interclass samples from the dataset which will be provided. This system will be more efficient for recognizing the real time traffic sign and also tell from which class a particular sign belongs with name. Therefore, this study makes TSR software has been developed by using Convolutional Neural Networks (CNN) built on DL techniques along with CV techniques. Coding is accomplished under TensorFlow and OpenCV frameworks with the python programming language and CNN training is carried out by using parallel architecture. The experimental findings indicate that the developed CNN architecture achieves greater accuracy and confirms the high efficiency of the system.

Keywords: *Deep Learning, Convolutional Neural Networks, traffic signs.*

1.INTRODUCTION

Traffic signs are warning, and caution signs are put on roads to advise drivers of road conditions and constraints or the way to go. Problems with road safety are mostly due to driver-specific subjective factors such as carelessness, inappropriate use of the driver, and non-compliance with traffic laws. For these reasons, lately autonomous vehicles have been a center of attraction for research study and development. And when we speak about autonomous vehicles Traffic Signs Recognition system is the first and foremost concept to include in any autonomous vehicles. Sometimes drivers may tend to miss the traffic signs along the route this can be dangerous and very unsafe with concern to road safety. In such cases automatic process of classification of traffic signs and reduce the number of road accidents on very large scale and can ensure complete safety. Many big companies in automation used this system in their cars using computer vision and machine learning approach but this was soon replaced by deep learning approach based on classifiers. Recently deep convolutional techniques have been proved to be the most effective for object detection. lt proves to be advantageous to look at the traffic signs recognition/classification with the deep learning the deep learning perspective. Classification of traffic signs is not a simple task it requires a huge dataset to go through various processes in deep learning. Initially the dataset gets divided into some ratio where those many images will go through one process and other no of images will go through other process and then will show the accuracy of the classes and then further it will get recognized. This concept has notable research work history and existing work going on still there are few areas/drawbacks that are yet to overcome. high-tech cars have recently become an efficient method for eliminating these human factors. TSR is the method of identifying traffic signs as automatic and it will provide the capability for smart cars and smart driving. Developing TSR systems that will enable state of the art vehicles to automatically recognize traffic signs such as speed limits, pedestrian crossings and inform the driver will provide extra security for drivers and passengers.

In summary, TSR is a 3-step process. The first one is that preprocessing is a process to reduce negative effects by arranging the pixels and dimensions in the images. The second one is the localization process that detects and localizes where a traffic sign is found in an image. Lastly, recognition is a process to recognize and classify the traffic sign by taking the localized traffic sign. In this study, software technology developed for detecting and recognizing traffic signs on a real-time based is explained.

2.LITERATURE REVIEW

Extensive study has been done in the area of recognition and classification of traffic and road signs. The authors proposed a Convolutional Neural Network and Support Vector Machines (CNN-SVM) method for traffic signs recognition and classification. The coloring used in this method is YCbCr color space which is input to the convolutional neural network to divide the color

channels and extracting some special characteristics. Their proposed method achieved a 98.6% accuracy for traffic signs recognition and classification. In another model the authors developed a new dataset consisting of 100,000 images and also proposed a traffic sign detection and classification method based on a robust end-to-end CNN. The method achieved 84% accuracy. The authors analyzed the spatial transformers and stochastic optimization methods for deep neural network for traffic sign recognition. They finalized this with a proposed system that achieved maximum accuracy.

There is a notable amount of research work done on traffic signs recognition/classification in history and still is going on. Different types of datasets have been used to solve different types of problems which includes detection, classification, tracking etc. After trying out and testing every object detection approach researcher got to the implementation with deep networks. In recent years with increasing technology and availability of standard datasets deep learning method is being more preferable. First CNN architecture ever used for traffic signs recognition was LeNet architecture.

In the paper 'Novel Deep Learning Model for Traffic Sign Detection Using Capsule Networks' by Amara Dinesh Kumar they had stated the various approaches been tried fpr traffic signs recognition in early stages. Also they have stated how CNN method is effective and preferable over other methods. They have also briefly stated about the capsule techniques.

In the paper 'Traffic Sign Classification Using Deep Inception Based Convolutional Networks' by Mrinal Haloi again they have explained the novel deep learning for traffic signs detection.

In paper 'A Novel Neural Network Model for Traffic Sign Detection and Recognition under Extreme Conditions' by Haifeng Wan they have mainly focused on the traffic signs detection by autonomous vehicles under extreme weather conditions.

In the paper 'Traffic Sign Detection for Intelligent Transportation Systems' by Ayoub Ellahyani they have survey regarding TSR system being installed in transportation vehicles and this paper proposes that it is a step towards intelligent transportation system.

In the paper 'Traffic Sign Detection and Recognition using Image Processing' by Karthikeyan D they have proposed the image processing approach towards TSR.

In the paper 'Two-stage traffic sign detection and recognition based on SVM and convolutional neural networks' by Ahmed Hechri. In this study, a novel two-stage approach for real-time traffic sign detection.

3.SYSTEM OVERVIEW

3.1. Software Development

In this section, the mathematical and scientific foundations of the subjects such as the used definition of the software development environment, the used programming language, the prepared database dataset classes, the used language libraries, and framework structures are explained. The development steps in general are very useful for developers to understand the logic and structure of this study.

3.2. Software Structure and Proposed System

In this study, the software is developed by installing the following packages on the phyton environment respectively; OpenCV, NumPy, Keras, Pandas, Scikit-learn, Scikit-image, TensorFlow for CPU and GPU. After the installation of these packages, the Python programming environment is created completely. GPU-based encodings accelerate the learning time and reduce the computational cost. The flowchart of the proposed study is illustrated in Figure 1. The collected dataset is used as an input to the CNN architecture for training, validation, testing, and classification.

3.3. Experimental Dataset

The dimension of the training range is an important aspect to take into account in DL learning models. The system becomes ambiguous without a sufficient number of training examples. In this study, public open-source datasets are used. Used TSR Benchmarks of the dataset. Traffic signs are manually created and tagged, in this way TSR data dataset is created. Dataset consists of 43 traffic sign classes and thousands of images. The low resolution and poor contrast of the images in the used dataset can reduce the success rate. In some cases, it can be difficult for the learning algorithm to recognize the traffic sign. Therefore, the better dataset is optimized and the higher the resolution images are used, the higher the success rate reaches of the TSR. The dataset used in this study is designed as multi-class, designed as a large and realistic database. The class distribution of the dataset is given in. The dataset is broken into sets for training, validation, and testing processes. This segmentation is used a statistical approach which is the Hold-out cross-validation method to measure and estimate the success rate of DL algorithms. This method is widely used because of its effectiveness and ease of application. As there are no set common instructions for the data set's percentage partitioning, commonly used partitioning percentages are used.

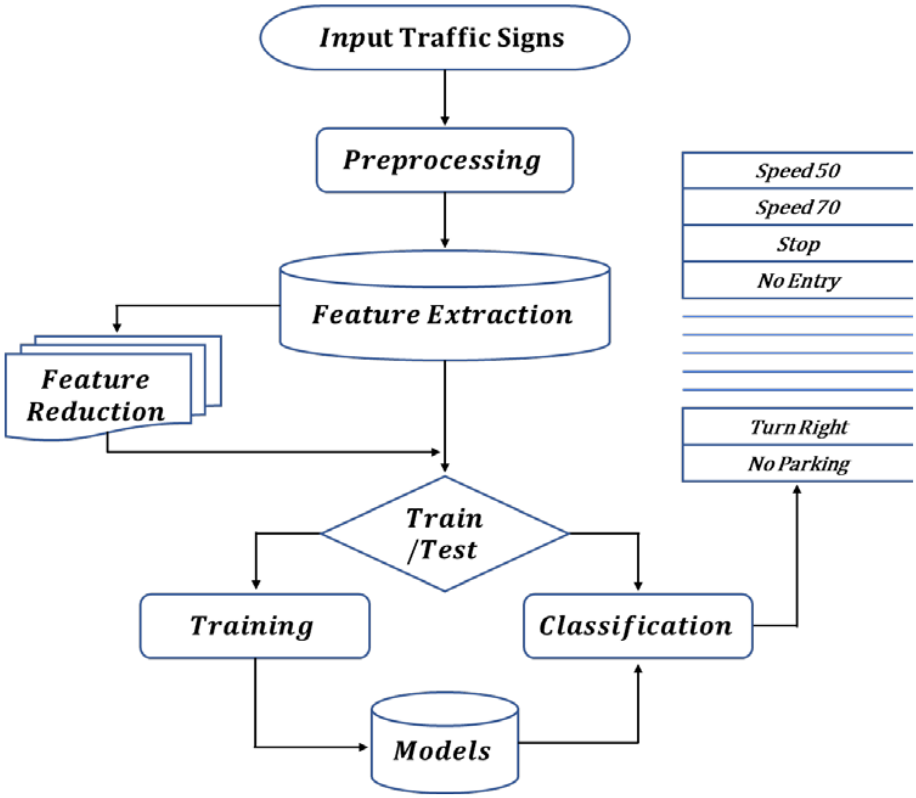


Figure 1. Flowchart of the proposed system

4.METHODOLOGY

In this section, the libraries, frameworks, algorithms, techniques, background, and fundamentals used in this study are explained and their contributions to the study are presented.

4.1. Computer Vision (CV) and OpenCV

OpenCV is a CV and DL application library that is open source, and it is designed to provide applications with common infrastructure and to accelerate the use of DL. CV libraries that contain algorithms such as image processing, pattern recognition, and histogram extraction are needed in the design and development of Real-Time systems. OpenCV includes open-source versions of these algorithms and OpenCV is preferred to be used in this study due to its open-source and no copyright issues.

4.2. TensorFlow Ecosystem

TensorFlow is not just a computational engine and a deep learning library for training neural networks. It is a complete ecosystem used (TensorFlow Lite for mobile and embedded devices) for the development of production machine learning pipelines and deploying-quantization production models. It includes sessions and willing execution, automatic differentiation, model and layer subclassification, and better multi-GPU/distributed training support functions. We can train, optimize, and quantify models designed to work on resource-constrained devices such as TensorFlow Lite, smartphones, and other embedded devices. Raspberry Pi and TensorFlow Lite were used in this study. The training and deployment steps of the TensorFlow ecosystem are shown in Figure 1.

4.3. Keras DL Framework

Keras is a high-developed deep learning framework for the phyton environment and Keras framework is used to create a deep CNN in this study. By using Keras libraries, a CNN is created and trained in the following steps: Loading data from disk, creating the training and the test sections, defining Keras model architecture, compiling Keras model, training the model with training data, evaluating the model over test data, and making predictions using the trained Keras model.

4.4. Convolutional Neural Networks (CNN)

We have used the LeNet architecture for the model here with few modifications. The model originally consists of total 7 layers. The layers consists of 3 convolutional layers ,2 subsampling layers and 2 fully connected layers. First layer is the input layer then there are 2 subsampling layers also known as maxpooling layer thereafter it has the fully connected layer and then lastly the output layer. So basically CNN has layer division as 3 types namely convolutional layers (Conv2D), pooling layers (Maxpooling2D) and fully connected layers. (fig.2)

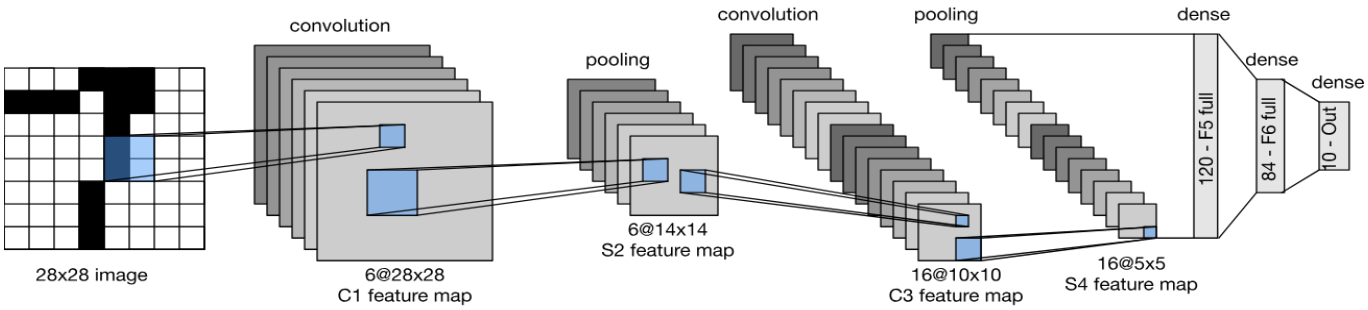


Figure 2. Standard LeNet Architecture

In fig.3 denotes the layers in a particular convolutional model. it takes the images as input then breaks down the images in small packets using the convolutional and pooling layers and then finally with the help of fully connected layers classification takes place and then we see the particular output.

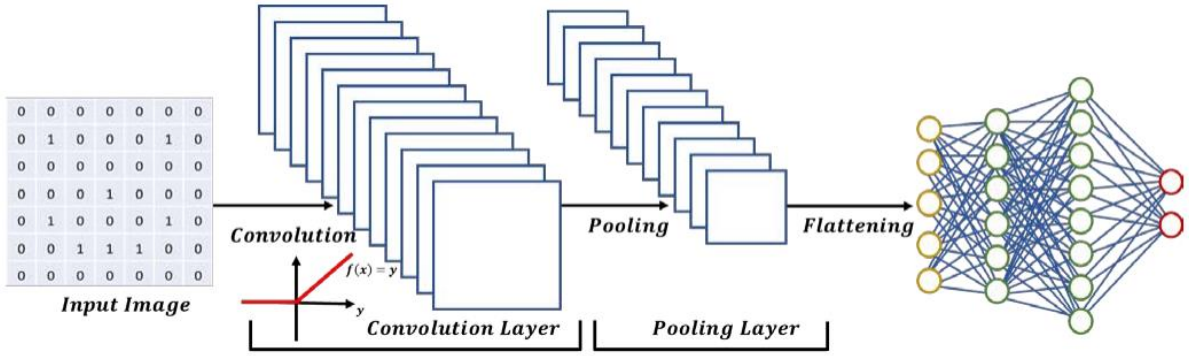


Figure 3. Entire process of creating and optimizing a CNN

In this particular project we have used LeNet architecture as mentioned before. In this model we have few convolutional layers, few pooling layers then we have few dropout layers and at the end we have a dense layer which is nothing but our output layer.

5.EXPERIMENTAL RESULTS AND DISCUSSIONS

The main.py script initiates a process to train a model using a dataset that comprises multiple classes of images. Initially, the script detects the number of classes present within the dataset, identifying a total of 43 distinct classes. Subsequently, it iterates through each class folder within the dataset, importing the images contained therein and aggregating them into a unified matrix representation. Each image is associated with its respective class label during this process.
Following the data aggregation step, the script proceeds to partition the dataset into distinct subsets for training, testing, and

```
Total Classes Detected: 43
Importing Classes.....
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42
```

```
Data Shapes
Train(25055, 32, 32, 3) (25055,)
Validation(6264, 32, 32, 3) (6264,)
Test(3480, 32, 32, 3) (3480,)
```

Figure 4. Splitting of dataset

validation purposes. This partitioning involves allocating a predetermined number of images to each subset, ensuring that the dataset is sufficiently representative for training, evaluation, and performance assessment. Specifically, a portion of the images is assigned to the training subset to facilitate the model learning process, while separate subsets are reserved for testing the model's generalization capabilities and validating its performance on unseen data.

Once the splitting is done it will show the distribution of the training dataset in graphical form. Studying this training distribution, it is important to know that we do not have same no of images for each class. So, as we can see in fig.4, we have about 100 images for first class and similarly we have about 1300 images for another class it clearly implies that the distribution is not even

so this may result in good classification for one class and bad classification for another class because the dataset is not evenly distributed for each class.

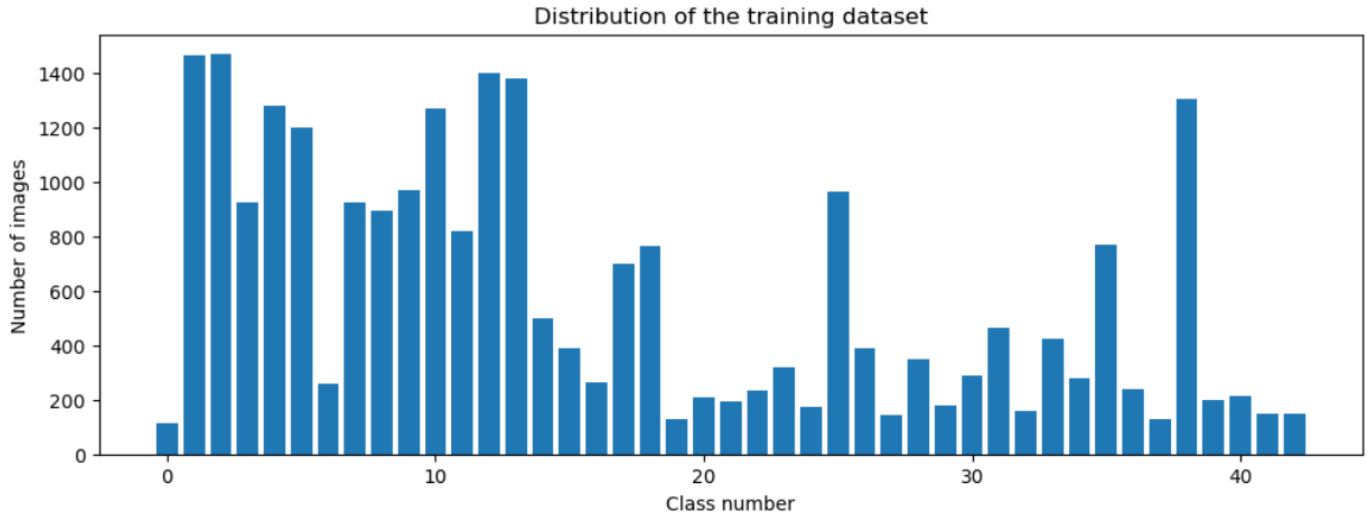
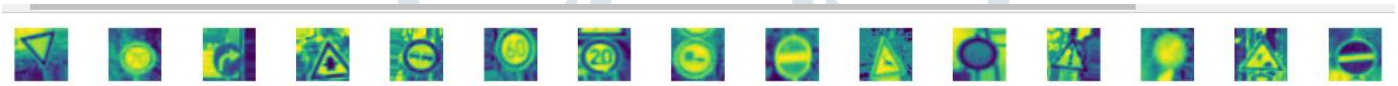


Figure 5. Training distribution

Once this is done the pre-processing of the images will take place that is the images will get converted to grayscale and it will show us example augmented image so that to Check that the preprocessing is done properly.



After compiling the model(fig.6) the training will start for the 10 epochs as parameter declared before and 502 steps per epochs. So, 2000 images will train for 10 iterations to give accuracy and loss of the model(fig.8).

```
None
Epoch 1/10
502/502 [==============================] - 0.7634
Epoch 2/10
502/502 [==============================] - 0.9125
Epoch 3/10
502/502 [==============================] - 0.9561
Epoch 4/10
502/502 [==============================] - 0.9625
Epoch 5/10
502/502 [==============================] - 0.9740
Epoch 6/10
502/502 [==============================] - 0.9746
Epoch 7/10
502/502 [==============================] - 0.9454
Epoch 8/10
502/502 [==============================] - 0.9757
Epoch 9/10
502/502 [==============================] - 0.9823
Epoch 10/10
502/502 [==============================] - 0.9866
```

```
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 (MaxPooling  (None, 4, 4, 30)         0
 2D)

 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
_________________________________________________________________
```

```
0.9285 - val_accuracy:
0.3295 - val_accuracy:
0.1799 - val_accuracy:
0.1216 - val_accuracy:
0.0957 - val_accuracy:
0.0880 - val_accuracy:
0.1601 - val_accuracy:
0.0737 - val_accuracy:
0.0674 - val_accuracy:
0.0556 - val_accuracy:
```

Figure 7. Training per epoch

In fig.8 accuracy, loss, validation accuracy is seen. So, the accuracy it gets is around 0.95 which is fair enough to classify the images i.e. it might get good results.

After that it plots the graph for accuracy and loss of the model it is getting fairly good results. For 10 epochs it is giving good results. After 4 epochs it is going at the same level. (fig.8) (fig.9)

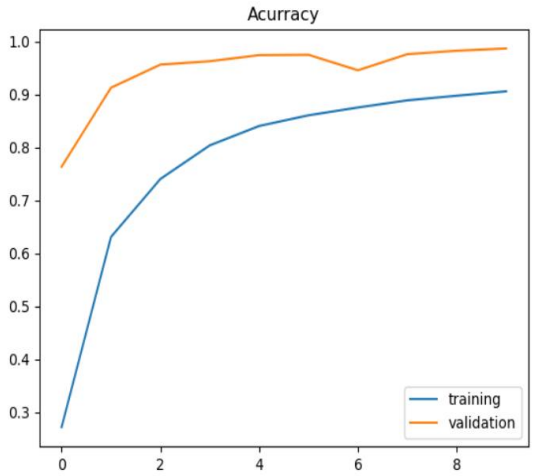


Figure 8. Accuracy plot

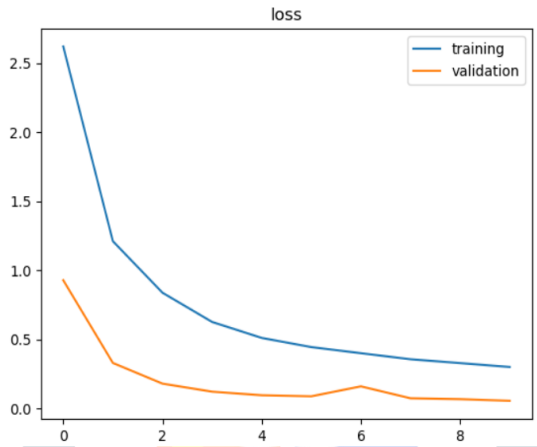


Figure 9. Loss plot

Once training is done when we give the path of an image it shows us the classification of the traffic signs using image processing techniques and will show the label and class of the particular traffic sign. (fig.10)

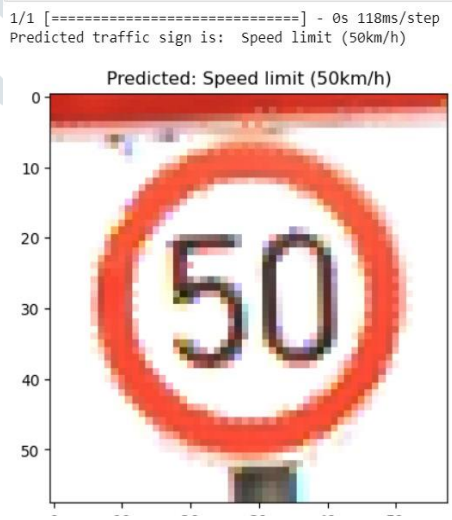


Figure 10. Image Classification

Once training is done it shows us the real time classification of the traffic signs using OpenCV and will show the labels, class and probability of how much percent the classification is true of the particular traffic sign. (fig.11)

Figure 11. Real time classification

5.CONCLUSION

From this proposed system we conclude that after building a efficient CNN model and training the dataset well it acquires good classification results. And gives good accuracy results as well. Originally to get good training distribution it requires the very big data sets but with the data sets that this system has used will also yield good results. In some existing autonomous vehicles, they have this system inbuilt but there could be many modifications and improvisations in the system to run this in every vehicle smoothly. Installing this system in vehicles can control, reduce upto 75% of accidents.

6.REFERENCES

[1] Mrinal Haloi, "Traffic Sign Classification Using Deep Inception Based Convolutional Networks",July 2016.

[2] Ayoub Ellahyani, llyas El jaafari and Said Charfi, "Traffic Sign Detection for Intelligent Transportation Systems: A Survey", E3S Web Conf. Volume 229, 2021.

[3] Lei Gao, Manman Su, Qinglong You, Hui Qu and Qirun Sun, "A Novel Neural Network Model for Traffic Sign Detection and Recognition under Extreme Conditions", July 2021.

[4] Karthikeyan D, Enitha C, Bharathi S, Durkadevi K, "Traffic Sign Detection and Recognition using Image Processing", vol.8, may 2020.

[5] C.-C. Jay Kuo., "Understanding Convolutional Neural Networks with A Mathematical Model", Sept 2020.

[6] Amara Dinesh Kumar, "'Novel Deep Learning Model for Traffic Sign Detection Using Capsule Networks" International Journal of Computer Applications. June 2011.

[7] Haifeng Wan, 'A Novel Neural Network Model for Traffic Sign Detection and Recognition under Extreme Conditions' , 2002.

[8] Wali SB, Hannan MA, Hussain A, and Samad SA. "Comparative Survey on Traffic Sign Detection and Recognition: A Review", Przeglad Elektrotech, 2015.

[9] Lina HY and Chang CC. "Traffic Sign Detection and Recognition for Driving Assistance System", AIVP-advance image and video processing, 2018.

[10] Shustanov A, and Yakimov P. "CNN Design for Real-Time Traffic Sign Recognition", ScienceDirect-Elsevier, 3rd International Conference-Information Technology and Nanotechnology, 2017.