Comparative Study of Deep Learning Approaches (CNN and VGG16) for Traffic Signs Recognition

Bandi Rupendra Reddyy1, Darukumalli Sai Tharun Reddy1, Sandeep Preetham M C1, Dr. Suja P.2, Dr. Peeta Basa Pati.2

1,2Department of Computer Science and Engineering, Amrita School of Engineering, Bengaluru,

Amrita Vishwa Vidyapeetham, India.

1[bl.en.u4aie19009@bl.students.amrita.edu](mailto:bl.en.u4aie19009@bl.students.amrita.edu), 1[bl.en.u4aie190016@bl.students.amrita.edu](mailto:bl.en.u4aie190016@bl.students.amrita.edu), 1[bl.en.u4aie19058@bl.students.amrita.edu](mailto:bl.en.u4aie19058@bl.students.amrita.edu), [2](mailto:5m_nithya@blr.amrita.edu) p\_suja@blr.amrita.edu, [2](mailto:5m_nithya@blr.amrita.edu) bp\_peeta@blr.amrita.edu,

*Abstract*—**Automobiles have evolved into the most functional mode of travel for every household in today's society. As a result, the road traffic environment is becoming increasingly complex, and people anticipate an intelligent Vision-assisted application that gives traffic sign information and governs driving operations. As one of the more crucial roles, traffic sign recognition has been a popular research topic among domestic and international experts. Due to their high recognition rate and quick execution, deep learning approaches for tackling classification problems have been highly popular in recent years. The majority of computer vision applications, both old and new, have been made better using convolutional neural networks. we propose an implementation of a traffic signs recognition algorithm that can classify traffic signs and learn and identify the most critical of these traffic signs features, with the goal of identifying traffic signs in the real world. In this project, training sets for recognizing traffic signs were created with the use of the German Traffic Sign Recognition Dataset and a convolutional neural network**

Keywords—**T*raffic sign recognition, Convolutional neural networks (CNN), GTSRB Dataset, Graphical user interface GUI.***

# Introduction

As both the population and the number of people who drive cars have increased, the roadway has become more complicated. This has led to an increase in the number of traffic accidents, which has led to an increase in the number of people who have been injured and the amount of property that has been damaged. In this case, traffic signs are a vital part of the overall safety of the people while driving on the roads. The main purpose of traffic signs is to display the contents that need to be noted in the present road sections and to alert vehicles in front of the road to the dangers and difficulties that may exist in the environment. Speed limits, prohibited entry, traffic signals, turning left or right, children crossing, no passing of big trucks, and so on are all examples of traffic signs. As a result, traffic sign detection and identification is an important research direction that is critical for minimizing road traffic accidents and protecting drivers' personal safety. The classification of traffic signs comes into play here because they offer drivers and passenger’s information that is essential to their safety. Traffic sign categorization refers to the process of identifying the particular class to which a certain traffic sign belongs.

Traffic Sign Classification is used to detect and classify traffic signs in order to advise and warn drivers ahead of time So that they do not break the rules. In the real world, the image of traffic signs is collected by the car's camera, allowing for real-time traffic sign recognition. However, there are a number of factors that influence traffic sign identification, including lighting factors, which affect image exposure, light weakness causes dim images, pictures taken while driving, which is blurred due to vibration, and traffic signs are visible all year. We used deep learning models in our project to overcome all of these challenges, such as corroded traffic signs.

The CNN extracts the features of the input photos on its own and learns the feature information of the photographs on a constant basis through training. It has more anti-interference features than artificial filtering functions. We created two-layer CNNs, three-layer CNNs, and a vgg-16 model for this project. Deep learning models can be used to create a traffic sign recognition system that can recognize and comprehend various traffic signs and assist the vehicles in which they are placed in making decisions and adhering to these laws. For picture categorization, these CNN models have given the best accuracies.

# Literature Survey

During the past decade, there has been a rise in the research and development of intelligent transportation systems, in particular vehicles that drive themselves. Finding a solution to one of these systems' most difficult challenges, which is identifying traffic signs, has been a challenge for both researchers and engineers. This problem is still being tackled as a problem of detecting, recognizing, and classifying objects (traffic signs) using computer vision.

The purpose of this research, which is presented in this study, is to recognize traffic signs, although it does not take into account the detection stage. This section exclusively discusses similar works from this perspective for this purpose. Extraction of features and recognition of the signs themselves are the two components that make up traffic sign recognition. During the initial stage, a number of different methodologies, including edge detection [1], scale invariance feature (SIFT) [2], Histogram of gradient (HOG) [4], and others, were put up as potential solutions.  Bag of Words (BOW) with SURF and k-means classifier was utilized in [5.] This step's output is typically used as input for classification algorithms that recognize road signs.

There have been many different approaches taken in the classification of traffic signs, some of which include the K-Nearest Neighbour (KNN) method, classifier [3], Support Vector Machine (SVM) [6], and neural network [5], [7]. Each road sign is stored with 200 characteristics. The Multilayer Perceptron Neural Network produces significantly superior outcomes. For several applications such as object categorization and pattern recognition, convolutional networks are gradually replacing classic computer vision techniques [7], [8]. It is used to extract further comprehensive descriptions of traffic signs and to learn those descriptions. The step of extracting descriptors, which can be highly unpredictable depending on a number of factors, is bypassed by using this method instead. A 2D image is sent into this network, and it performs a convolutional operation on it. It is possible to learn a visual representation that is representative of something.

# Methodology

## Dataset

The dataset we used in our project is the German Traffic Sign Recognition Dataset (GTSRB). It comprises over 40,000 images of various traffic signs that are further divided into 43 separate groups of signs of varied sizes, lighting circumstances, and occlusions, and is highly comparable to real-life data. The dataset is divided into two files called to train and test, each of which contains photos sorted into 43 groups and used to train our model. The test folder, on the other hand, contains photos of traffic signs in various circumstances that are used to test the model. Each of the existing classes has its own folder in the training dataset. There are total of 39209 tagged photos in the training set and 12630 images in the test set. There's also a CSV file with the path of each image and its class, as well as other information like width, height, file name, size, coordinates of the bounding box where the sign must be discovered, and the class label applied.

## Data Preprocessing

Images need to be converted into NumPy arrays (i.e. numeric values) to perform image processing. After loading the images in the dataset, they are resized to 30\*30 pixels. Post this, the labels of the image are mapped with the image and hence the dataset is ready to be trained.

## Models

CNN model:

The term "CNN" stands for "Convolutional Neural Network," which is an algorithm of Deep Learning. CNN can take a photo as input, give priority to various elements in the picture, and differentiate them from one another. It requires less preparation than other classification algorithms, especially when compared to those other classification methods. In contrast to the simple approaches to filters, which are done manually, the Convolutional Network has the ability to learn the filters or characteristics in the images.

The following steps are used to construct the CNN model architecture:

* The next layers should be added in the following order: two convolutional layers, one pooling layer, a dropout layer, a flattening layer, a dense layer, again a dropout layer, and finally the dense layer.
* The number of convolutional layer filters has already been decided upon. Convolution is applied to the source image before it is used in the process of creating a feature map.
* A rectified feature map is produced as a result of the ReLU's utilization of the maximum function, which converts negative values to zero while preserving the integrity of positive values. The Pooling layer is responsible for executing a down sampling operation on the image in order to reduce its dimensionality. This process is performed using the rectified feature map (such as Max Pooling or Average Pooling).
* Through the utilization of the flattening layer, the input feature map is transformed into a one-dimensional array.
* During the training phase, some of the input neurons will have their weights set to 0 thanks to the dropout layer, which helps prevent overfitting. The dense layer, on the other hand, is responsible for feeding all of the outputs from the layer below it to all of its neurons. Additionally, it is responsible for performing the matrix-vector multiplication, which results in the generation of an m-dimensional vector (the row vector of the output from the layer below should be equal to the column vector of the dense layer).
* After adding the layers, you should compile the model (the final stage in the development of a model, which is the stage in which you define the loss function and utilize optimization techniques), and then use the "Adam optimizer" with the loss function set to "sparse categorical cross-entropy." The proposed method addresses a multiclass classification problem, which indicates that a number of different classes are taken into consideration, but that each image only corresponds to a single class.
* After then, the model is trained to ignore the pre-processed photographs that are included in the training dataset and go straight to the raw data. In the last step, the trained model is applied to the test data in order to generate predictions, and the output displays the class Id in addition to the name of the traffic sign.

VGG-16 Model:

A 16-layer deep convolutional neural network is represented by the VGG-16. You have the option to import a pre-trained version of the network from the ImageNet database, which includes a version that has been trained on more than one million photographs. The network is able to categories photographs into one thousand distinct object categories, some of which include animals, keyboards, mouse, and pencils, among other things. Because of this, the network has learned a wide range of different feature representations that are rich for a variety of different images. The VGG model investigates layer depth using convolutional filter sizes that are extremely small in order to handle images of a vast scale (3x3). Thirteen convolutional layers, five max-pooling layers, and three fully linked layers are included in VGG16's architecture. As a consequence of this, there are sixteen layers, each of which has adjustable settings (13 convolutional layers and 3 fully connected layers). Because of this, the model has been given the moniker VGG16. The number of filters in each succeeding block is increased by a factor of two until the total number of filters reaches 512. The number of filters that are contained in the first block is 64. This model is complete with two hidden layers that are fully coupled to each other as well as one output layer. The total number of neurons in both of the layers that are fully connected is the same, which is 4096. The number of categories present in the Imagenet dataset is 1000, which is the same as the number of neurons present in the output layer, which is also 1000.

## Implementation

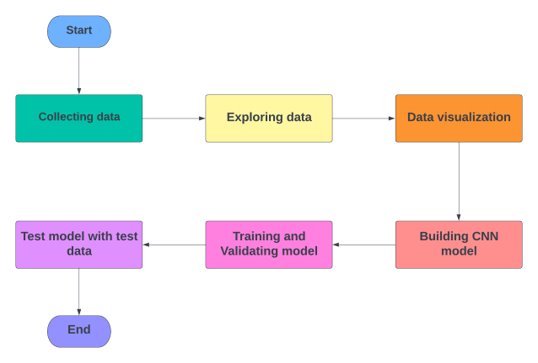


Fig. 1. Model Architecture

It is necessary for us to accomplish the following four steps in order to build our traffic sign classification model: dataset investigation, development of a CNN model with two and three layers, and construction of a VGG-16 model. In addition to model testing, we also do model validation and training.

## Dataset exploration:

In our 'train' folder, there are around 43 subfolders (numbered 0 to 42), each representing a distinct class. We have an OS module that assists in iterating all of the photos with their associated classes and labels. The PIL library is used to open the contents of ideas into an array. Finally, we must create lists for each image and its associated labels. To feed the data to the model, we need a NumPy array, so we transform this list into an array. The current structure of our data looks like this: (39209, 30-30-3), where 39209 stands for the total number of images, 30-30-3 for the image sizes in pixels, and 3-3-3 for the RGB value (availability of colored data). Before we begin developing the model, we need to first segment our dataset into a train set and a test set. We have reserved twenty percent of the data for testing reasons, in addition to using the remaining eighty percent for instructional purposes. The labels are then encoded using a single hot encoding after that process is complete. It is always a good idea to look at a few samples before beginning to work with images. Let's examine 25 images at random from the collection in order to determine the labels they have attached to them. In addition to this, we use sorted grids to visually represent the training datasets.

## Building the models:

## CNN(2-Layer) model building:

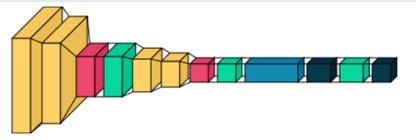


Fig. 2. CNN 2 Layer model

Two convolutional layers, one fully connected layer, and an output layer make up our very own custom model, which is referred to as a CNN 2 layer.

Convolutional layer:

Let's say we have an image that's 6 pixels on each side. It is necessary to define a weight matrix in order to extract certain features from the photographs. The weight has been set up as a 3\*3 matrix. This weight will now be applied to the image in such a way that it covers all of the pixels at least once, resulting in a convolved output. The 6\*6 image has now been reduced to a 4\*4 size. Consider the weight matrix as a paintbrush on a wall. The brush paints the horizontal portion of the wall first, then descends to paint the next row horizontally. When the weight matrix travels along with the image, pixel values are used again. In a convolutional neural network, this facilitates parameter sharing.

Consequently, we are making use of the padding in order to maintain the same output after the convolution layer. In our project, the stride is set to 3, and the padding is set to 1, so the output and input sizes will be the same. After that, we employ max pooling to cut down on the number of dimensions, and we use dropout to get rid of unneeded features in order to cut down on overfitting. After training with the softmax activation function to generate probabilities, the same procedure iterates continuously for two layers, and then the output is sent to a layer that is fully connected. We are able to categorize the traffic signals according to their pictures by using the probabilities that are most likely to occur.

CNN (3-Layer) model building:

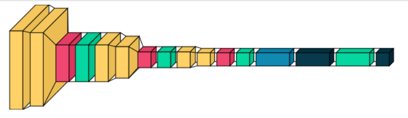


Fig. 3. CNN 3 Layer model

Three convolutional layers, one fully connected layer, and one output layer are included in our own model, which is a CNN with three layers.

Similar to the CNN  2 layer same structure follows in this model. But it consists of one more convolutional layer.  By comparing these models we got better accuracy for the two-layer CNN model rather than 3 layer CNN model. From this, we have concluded that by increasing the no of layers test accuracy is increased and it leads to overfitting.

.

*VGG-16 model building:*

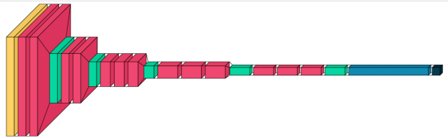


Fig .4. VGG16 Layer model

The first two stacks each include a maximum pooling layer that is 2 x 2 and has a stride of 2, followed by two consecutive convolutional layers that each has 64 filters that are size 3 x 3. In addition, the size of the supplied image cannot be changed from 224 by 224 by 3. Following that, we build the other layers while adhering to the specifications outlined in the architectural section. Following these layers of convolutional stacks, we need to add three layers that are fully connected. The very top layer is going to be the output layer, and it will use Softmax activation. Before we add the first layer that is completely connected, we need to add a layer that has been flattened. For the purpose of regularisation, we shall apply dropout layers directly after thick layers. It is possible to modify the size of the output layer so that it corresponds appropriately to the number of classes in your recognition challenge.

## Model training and validation:

To train our model, we'll utilize the model.fit() method, which works well when the model architecture has been successfully built. We achieved 94 percent accuracy on training sets and achieved stability after 20 epochs using 64 batch sizes. After that, we plot graphs for accuracy.

## Model testing:

Our dataset contains a subdirectory named" test," which contains the main working comma-separated file" test.csv." It consists of two elements: picture paths and their associated class labels. To extract the image path with related labels, we can use the panda's python library. To forecast the model, we must scale our photographs to 30x30 pixels and construct a NumPy array loaded with image data. Importing accuracy score from the sklearn.metrics package will help us analyse how the model predicts the actual labels. Finally, we use Keras' model.save() method to store our trained model.

## GUI for Traffic Signs Classifier:

To create a graphical user interface (GUI) for our traffic sign recognizer, we'll utilize Tkinter, a standard Python package. For this, we'll need to make a separate python file called "gui.py." To begin, we'll use the Keras library's deep learning technique to import our trained model 'traffic classifier.h5.' Following that, we create a user interface for uploading photographs as well as a classifier button to identify which class our image belongs to. Because of this, we need to write the classify () method, which is automatically invoked whenever we click on the GUI button. In order to anticipate the traffic sign, we will need to use the identical shape resolutions that we utilized when the model was being trained. As a consequence of this, within the classify () method, we transform the image to the shape dimension, which is equal to one hundred thirty times thirty times three. The model predict classes (image) method is used for image prediction. This method, which returns the class number (0-42) for each picture, is called. Once we have this class number, we can access the dictionary and pull out the relevant information.

# Results

In this project, we have used CNN 2, CNN 3 Layers, and VGG16 deep learning methods as the classifier to perform the traffic sign recognition. We evaluate the performance of CNN and VGG16 models by using the accuracies method and compare the recognition performance of the three deep learning models.

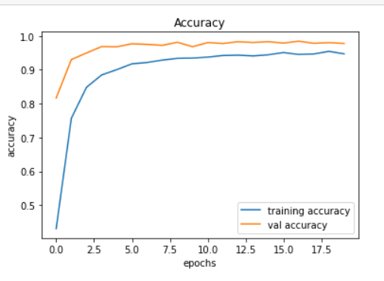


Fig. 5. Accuracy curve of CNN (2- Layers) model on GTSRB.

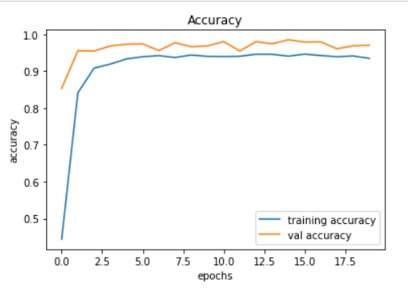


Fig. 6. Accuracy curve of CNN (3- Layers) model on GTSRB.

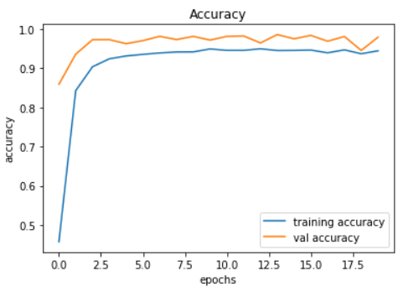


Fig. 7. Accuracy curve of VGG-16 model on GTSRB.

The accuracy curves of the three models are shown on the GTSRB data set in the figures that have been presented above. The accuracy of GTSRB on CNN (2-Layers) can reach 98.35%, which is having high accuracy than the two models, then CNN (3-Layers) can reach 97.35% and the VGG-16 model can reach 97%..

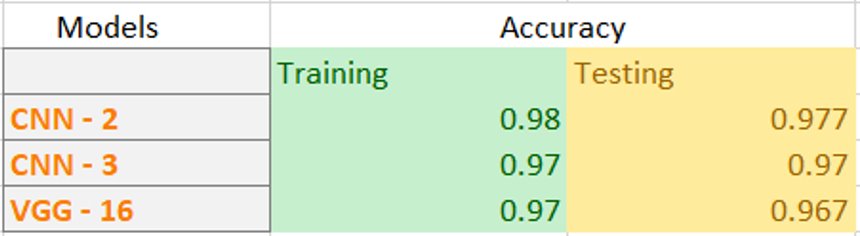


Table. 1. Training and testing accuracies for 3 models

From the table, we can see that the CNN (2 Layers model) is getting both training and testing greater than the other two models, CNN (3 Layer) model and the VGG-16 model for the given GTSRB Dataset.

When displaying the results of our models built using Tkinter, we turned to Gui for help. Out of all the several GUI techniques available, the one that is utilized the most is called Tkinter. It's an interface for the Tk GUI toolkit that comes along with Python, and it's written in Python. We do testing for photos that include not just traffic signs but also other types of signs, such as human photographs and a great many more.

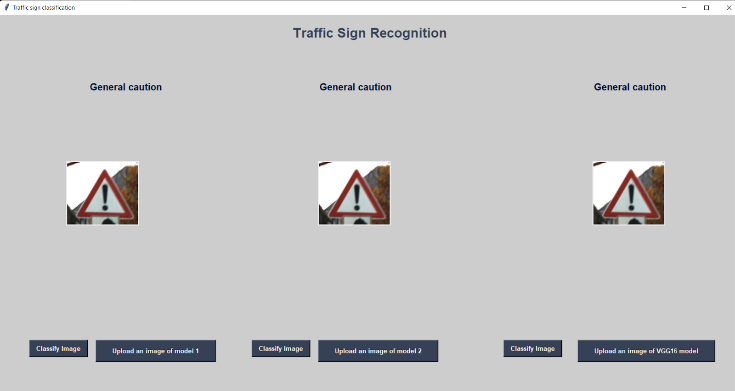


Fig. 8. Sample test image1 from dataset.

This is one of the test cases, here we are given the same traffic sign image and our model is predicted correctly.



Fig. 9.Sample test images from dataset.

Here we are given different images for different models and it predicted correctly.

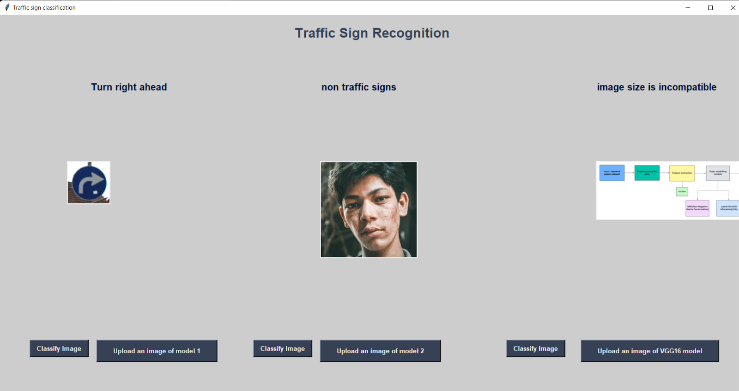


Fig. 10. Random test images from internet.

Here we are given non-traffic sign images which are a human image and a random image.it predicts the non-image correctly and we have introduced the size incapability for our model if the image is larger than the excepted image then it will the size incapability popup.

# Conclusion

Within the scope of this study, we utilized a variety of deep learning models in order to demonstrate and develop an identification system for traffic signs. We were able to successfully classify the traffic signs classifier using three different models of accuracies in this project, with 96.7 percent for CNN (2 Layers), 96.7 percent for CNN (3 Layers), and 96.7 percent for VGG-16 on the German Traffic Sign Recognition Dataset. However, we did observe variations in accuracy and loss. While working on this project, we made use of this model's graphical user interface, which makes it easy to see how different sorts of signs are categorized. CNN is effective at recognition, and by adjusting the hyper-parameters, one may improve both the accuracy and recognition rate of the algorithm. As a consequence of this, we incorporated CNN into the proposed plan for recognizing traffic signs in order to develop a system that allows drivers to recognize traffic signs. The images will be taken with a camera that is attached to the vehicle while the picture acquisition stage is being performed, and the CNN algorithm will be used to perform the recognition after the images have been preprocessed.

##### References

[1] P. Dewan, R. Vig, N. Shukla and B. K. Das, "An Overview of Traffic Signs Recognition Methods", *International Journal of Computer Applications*, vol. 168, no. 11, June 2017.

[2]  D. Jianmin and V. Malichenko, "Real-time road edges detection and road signs recognition", *IEEE International Conference on Control Automation and Information Sciences (ICCAIS)*, 29-31 Oct. 2015.

[3] Y. Han, K. Virupakshappa, E. Vitor, S. Pinto, and E. Oruklu, "Hardware/Software Co-Design of a Traffic Sign Recognition System Using Zynq FPGAs", *In Electronics journal*, vol. 4, pp. 1062-1089, 2015.

[4] F. Zaklouta and B. Stanciulescu, "Real-Time Traffic-Sign Recognition Using Tree Classifiers", *IEEE Transactions On Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1507-1514, December 2012.

[5] K. Tohidul Islam, R. Gopal Raj, and G. Mujtaba, "Recognition of Traffic Sign Based on Bag-of-Words and Artificial Neural Network", *Symmetry journal*, vol. 9, pp. 138, 2017.

[6] S. Maldonado-Bascón, S. Lafuente-Arroyo, P. Gil-Jiménez, H. GómezMoreno and F. López-Ferreras, "Road-Sign Detection and Recognition Based on Support Vector Machines", *IEEE Transactions On Intelligent Transportation Systems*, vol. 8, no. 2, pp. 264-278, June 2007.

[7] L. Abdi, "Deep learning traffic sign detection recognition and augmentation", *Proceedings of the Symposium on Applied Computing*, pp. 131-136, 2017.

[8] Y. Moualek, "Deep learning pour la classification des images", *Master’s thesis*, 2017.

[9] S. Reddy, D. Dash, and N. Rakesh, "Image Classification Using Machine Learning Techniques for Traffic Signal" in Intelligent Data Communication Technologies and Internet of Things, Singapore: Springer, pp. 233-244, 2021.

[10]  A. Shustanov and P. Yakimov, "CNN design for real-time traffic sign recognition", *Procediaengineering*, vol. 201, pp. 718-725, 2017.

[11]  M. Mathias, R. Timofte, R. Benenson and L. Van Gool, "Traffic sign recognition - how far are we from the solution?", *Neural Networks (IJCNN) The 2013 International Joint Conference on*, pp. 1-8, Aug 2013.

[12] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "Imagenet classification with deep convolutional neural networks", *Advances in neural information processing systems*, pp. 1097-1105, 2012.

[13] F. Zaklouta, B. Stanciulescu, and O. Hamdoun, "Traffic sign classification using k-d trees and random forests", *Neural Networks (IJCNN) The 2011 International Joint Conference on*, pp. 2151-2155, July 2011.

[14] J. Stallkamp, M. Schlipsing, J. Salmen and C. Igel, "The German traffic sign recognition benchmark: a multi-class classification competition", *The 2011 international joint conference on neural networks*, pp. 1453-1460, 2011, July.

[15] Q. Zheng and X. Xie, "Traffic Sign Recognition Based on Learning Vector Quantization and Convolution Neural Network", *Proceedings of the 3rd International Conference on Intelligent Information Processing*, pp. 178-183, 2018, May.

[16] P. Dhar, M.Z. Abedin, T. Biswas, and A. Datta, "Traffic sign detection — A new approach and recognition using convolution neural network", *2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, pp. 416-419, 2017.

[17] Shustanov, P. Yakimov, “CNN Design for Real-Time Traffic Sign Recognition,” 3rd International Conference “Information Technology and Nanotechnology,” ITNT- 2017, 25-27 April 2017, Samara, Russia.