WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Traffic Symbols Detection Using Artificial

Dr. Ranjan Kumarmahapatra, A.S.Gayazuddin, S.Ashiq Basha S.Baba Shohail

Department of Electronics and Communication EngineeringMadanapalle Institute Of Technology And Science Madanapalle, Andhra Pradesh-517325
Email: ranjankumarm@mits.ac.in,18691A0443@mits.ac.in,18691A0416@mits.ac.in,18691A0413@mits.ac.in

ABSTRACT

The life on earth is turning out to be so digitalized with the most recent advancements like AI, IOT, Deep learning and so on. Man-made reasoning (AI) is essentially the investigation of preparing your machine (PCs) to mirror a human cerebrum and its reasoning capacities. The world is speeding up at higher speed in the areas of AI, profound learning, mechanical technology and so forth as we know everything is computerized in this day and coming ages. As a piece of sub-area, the presentation of Convolution Neural Networks made deep advancing broadly solid in the space of picture characterization and location. The examination that we have directed is one of its sorts.Throughout the long term, traffic sign location and acknowledgment frameworks give additional worth to driver, help when

INTRODUCTION

Previously and late occasions, there have been numerous street mishaps where the primary justification for these being lacking driving, prompting easier to use driving experience and significantly better wellbeing for travelers. As a component of Advanced Driving Systems (ADAS) one can be benefitted by utilizing this framework particularly with driving insufficiencies by alarming and help them about the presence of traffic signs to limit undesirable conditions during driving like exhaustion, helpless sight and antagonistic climate conditions. However, a different number of traffic sign recognition frameworks have been overhauled in writing; the need of plan with a powerful calculation actually stays open for additional examination. This paper proposes to plan a framework fit for performing traffic sign identification while considering varieties of difficulties like shading brightening, computational trouble and useful requirements existed.discovered that the second most heard explanation was an individual not knowing what a specific traffic sign implied. We unequivocally accept that the exploration that we have done would assist people with learning these signs naturally, particularly the puberty of 21st century, who likewise stay and live around innovation, which is becoming quicker than at any other time. Our exploration centers around recognizing traffic signs, when given a picture to it through profound learning, picture handling through OpenCV and a helpful UI is having been created in Python GUI utilizing Tkinter library. Nowadays, there is a lot of attention being given to the ability of the car to drive itself. One of the important aspects for a self-driving car is the ability for it to detect traffic signs in order to provide safety and security for the people not only inside the car but also outside of it. There are many various types of traffic signs, such as speed limits, traffic lights, designating directions (left or right), and so on. The task may be stated simply as follows: categorize the types of traffic signs. The main objective is to design and construct a computer-based system which can automatically detect the road signs so as to provide assistance to the user or the machine so that they can take appropriate actions. The proposed approach consists of building a model using convolutional neural networksby extracting traffic signs from an image usingcolor information. I will be using convolutional neural networks (CNN) to classify the traffic signs.

The fundamental goal of this exploration is to foster an item which would assist individuals with finding out around one of the most underestimated, yet particularly important part of our regular routine, a traffic sign. This model has been made utilizing profound learning libraries TensorFlow and its significant level API (Applications Programming Interface), Keras. The goal of this model is to achieve an exactness so solid that an individual ought to have the option to utilize our item without a second thought. So, by developing anefficient traffic sign detection system we could be able to help with different daily life applications.

The greatest imperative that we had throughout this undertaking was the need of computational power that was expected to prepare and test the informational collection. With an absolute size of more than 50000 pictures, this model requires very some measure of computational ability to prepare the organization similarly quicker, and seeing as such high computational power isn't available effectively. The second imperative we confronted were the moving of pictures from the downloaded informationalcollection to preparing and testing sets, which additionally requires a seriously decent measure of computational power.Traffic sign detection and recognition plays an important role in expert systems, such as traffic assistance driving systems and automatic driving systems. It instantly assistsdrivers or automatic driving systems in detecting and recognizing traffic signs effectively.

MODEL DESIGN ANDANALYSIS

- **DATASETS**

  The GTSRB-German traffic sign detection benchmark dataset was utilized in this study. This is a widely used dataset for traffic signals

on platforms such as Kaggle. There are over 43 picture classes in this dataset, with over 5000 photos for testing and validation. We created a mini data of ten classes from this dataset for testing, training, and validation. so that we can see how the machine is learning Our dataset was further separated into three sections: testing, training, and validation, which allows us to assess how well our model is performing.

Fig:2.1.a Dataset images The figure 2.1.a shows some of thetraffic sign board images from thedataset.

It is evident from the above numbers that this dataset has been meticulously produced so that it may be utilized for future study. Both training and testing were carried out on a workstation with an i5-9th Gen processor. The workstation was equipped with 16 GB of RAM and a 512 GB solid-state drive. Because we had access to an Nvidia GTX 1080, we were able to create a more efficient computing environment for both training and testing our deep learning model.

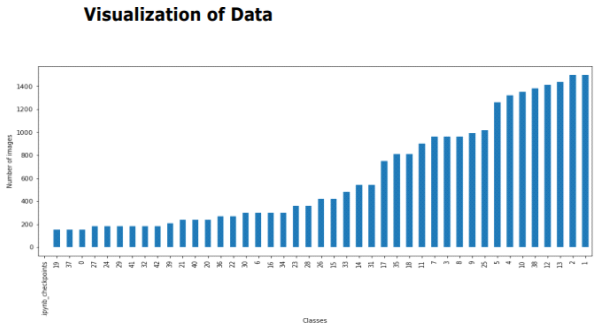


Fig:2.1.b visualization of dataset The figure 2.1.b is a data visualizationgraph for classes vs number of imagesin the dataset.

- **SOFTWARE REQUIREMENTSPECIFICATIONS**

GOOGLE COLAB:

Collaboratory, or "Colab" for short, is aproduct from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.We have used multiple dependencies for our research. As this research requires manipulation as well as visualization of the data being used, there was extensive use of libraries.

The main dependencies we used for ourresearch are:

- Tensor flow
- Keras 3.Numpy

- Pandas 5.Opencv 6.Matplotlib 7.Tkinter 8.Sci-kit learn
-

CONTENT DIAGRAM OF MODEL

The primary building block of a CNN is a convolutional layer. It includes a collection of filters (or kernels) whose parameters must be learnt during the course of the training. The filters are often smaller in size than the real image. Each filter builds an activation map by combining the picture with it. For convolution, the filter is slid over the image's height and breadth, and the dot product between each filter element and the input is calculated at each spatial point. The convolution process is illustrated in Fig. By convolving the filter with the blue component of the input picture, the first item of the activation map (marked blue in Fig) is computed. This technique is repeated for each element of the input picture to build the activation map. The activation maps of each filterare stacked along the depth dimensionto form the convolutional layer's output volume.

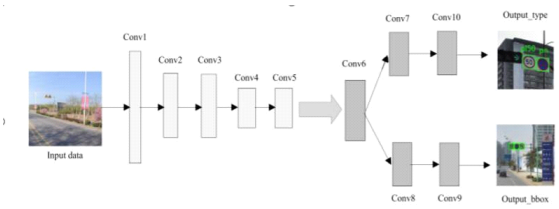


Fig:2.3 Content Diagram

The figure 2.3 explains the basic convolutional layers involved in CNN.Every component of the activation map may be considered a neuron's output. As a result, each neuron is linked to a small local region in the input picture, the size of which is equal to the filter's size. The parameters of all the neurons in an activation map are shared. The network is pushed to train filters that have the best response to a particular region of the input due to the convolutional layer's local connection.

- **IMPLEMENTATION**

Traffic signs can be dissected utilizing forward aligned cameras in numerous cutting-edge vehicles, vehicles and trucks. One of the fundamental use instances of a traffic-sign acknowledgment framework is for speed limits. The greater part of the GPS information would acquire speed data, yet extra speed limit traffic signs can likewise be utilized to separate data and show it in the dashboard of the vehicle to caution the driverabout the street sign. This is a high-level driver-help include accessible in most very good quality vehicles, essentially in European vehicles.A model calculation for traffic-sign location current traffic-sign acknowledgment frameworks is being created utilizing convolutional neural organizations, predominantly determined by the necessities of independent vehicles and self-driving vehicles. In these situations, the discovery framework needs to distinguish an assortment of traffic signs and not simply speed limits. A convolutional neural organization can be prepared to take in these predefined traffic signs and 'master' utilizing Deep Learning strategies.

FUNCTION IMPLEMENTATION There are some key parameters used in ourresearch which helped in better training ofthe model.

- Activation functions
- optimizers
- back normalization
- max pooling

Activation Functions

The activation function calculates a weighted total and then adds bias to it to determine whether a neuron shouldbe activated or not. The activation function's goal is to induce non- linearity into a neuron's output.

Optimizers

Optimizers are used to fine-tune a model's parameters. The goal of an optimizer is to maximize a loss function by adjusting model weights. The function is used to assess how effectively the model is working.

Adam Optimizer:

Adaptive Moment Estimation is an approach for gradient optimization that uses an gradient decent algorithm. When dealing with huge problems with many variables, the approach is quite effective.

Batch Normalization

overfitting is one of the most serious issues that arise when working on dnn projects. Regularization techniques aid in the improvement of a model and enable it to converge. Normalization is a data pre-processing technique for converting numerical data to a common scale without changing the form of the data. When we feed data into a machine learning or deep learning system, we usually modify the numbers to a balanced scale. Normalization is done

in part to guarantee that our model can generalize correctly.Batch normalization is a method of making neural networks quicker and more stable by adding extra layers to adeep neural network.

Max pooling

In Convolutional Neural Networks, Max Pooling is a down sampling approach. Max Pooling is a pooling technique that computes the maximum value for a feature map's patches and utilizes it to produce a down sampled (pooled) feature map. It is often employed following a convolutional layer. It adds a little bit of translation invariance, which means that translating the picture by a small amount has little effect on the values of most pooledoutputs.

DATA AND FEATURE GENERATION

Features are the patterns in your data set that can be used to train models. Good features (which we'll learn to identify in a moment) can help you to increase the accuracy of your Machine Learning model when predicting or making decisions. Our data set will have many features, but not all are relevant.With feature selection, you can avoid wasting time calculating and collecting useless patterns that you'll have to remove later.

Feature selection helps you simplify your ML models and enables faster, more effective training by:

- Removing unused data.
- Avoiding "Garbage in Garbage Out".
- Reducing overfitting.
- Improving the accuracy of a MachineLearning model.
- Avoiding the curse of dimensionality.

DATA PREPROCESSING ANDFEATURE SELECTION

We have used 43 different types of traffic signs. Initially, open the directory containing the dataset and resized each image. Each image is added to the data list, and the associated classes are added to the labels list. We now make numpy arrays out of these lists. Each image is added to the data list, and the associated classes are added to the labels list. The dataset was then divided into two parts: 80 percent for training and 20 percent for testing. The class labels for the train and test datasets are now converted into a single hot encoding into a 43x43 vector representation using to categorical().

DATA PARTIONING, LEARNINGAND EVALUATION

Data partitioning is normally used in supervised learning techniques in data mining where a predictive model is chosen from a set of models, using their performance on the training set as the validation of choice.So, the data is converted into three small data models. They are: Training Data, Testing Data and Validating Data.

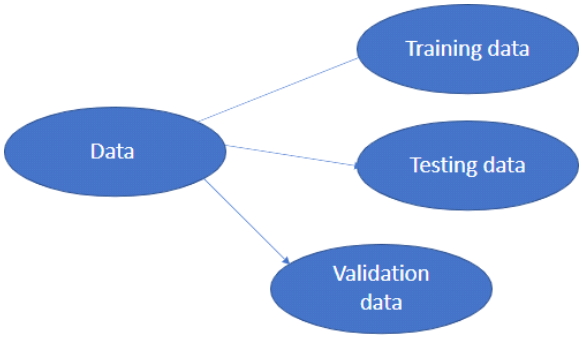


Fig:3.4 Data model

The Fig 3.4 describes the data portioning involved in the data model

TESTING AND RESULTS

- **CONVOLUTIONAL MODEL**

The model I created is a sequential model. Conv2D generates a 2D convolutional kernal with a kernel size of (5,5) (the dimensions of the kernel for processing the image) and an activation function of Rectified Linear Unit which is a max function which sets all negative values to zero and other values constant as arguments. Maxpool2D reduces image dimensions, and the pool size I have used is (2,2), which corresponds to a 2x2 window. Dropout is used to ignore some neurons (data points) from the dataset atrandom while fitting the model. It was used todeal with model overfitting.

To convert a ND array to a 1D array, I used Flatten (). To create a multiclass classifier, the activation function softmax is used. The model is compiled using categorical cross entropy as the loss function. The categorical cross entropy function is used to reduce the distance between the predicted and actual class label. Adam, the optimizer, is used to determine the individual learning rate for each parameter. An epoch is the number of iterations required to fit the model. In a batchsize of 32 images, the model is fitted for the trained dataset.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 30, 30, 3)]       0

 conv2d (Conv2D)             (None, 30, 30, 32)        2432

 conv2d_1 (Conv2D)           (None, 30, 30, 32)        25632

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 dropout (Dropout)           (None, 15, 15, 32)        0

 conv2d_2 (Conv2D)           (None, 15, 15, 64)        18496

 conv2d_3 (Conv2D)           (None, 15, 15, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 64)         0
 2D)

 dropout_1 (Dropout)         (None, 7, 7, 64)          0

 flatten (Flatten)           (None, 3136)              0
```

- **PERFORMANCE PARAMETERS**

The performance parameters are considered for the validation process.

There are mainly two parameters to consider and compare. They are Loss value and Accuracy. Training of the model is completed successfully. It is observed that loss value is reduced and the validation accuracy and accuracy graphs are at its high efficiency finally obtained values

Loss value:0.0759 Accuracy:0.976 Val_Loss:0.0034 Val_accuracy:1.000

```
1 - val_loss: 0.0146 - val_accuracy: 0.9978
Epoch 19/25
66/66 [==============================] - 14s 209ms/step - loss: 0.1073 - accuracy: 0.962
9 - val_loss: 0.0161 - val_accuracy: 0.9956
Epoch 20/25
66/66 [==============================] - 14s 210ms/step - loss: 0.0695 - accuracy: 0.974
8 - val_loss: 0.0065 - val_accuracy: 0.9978
Epoch 21/25
66/66 [==============================] - 14s 220ms/step - loss: 0.0740 - accuracy: 0.976
2 - val_loss: 0.0103 - val_accuracy: 0.9956
Epoch 22/25
66/66 [==============================] - 14s 208ms/step - loss: 0.0410 - accuracy: 0.988
6 - val_loss: 0.0047 - val_accuracy: 0.9967
Epoch 23/25
66/66 [==============================] - 14s 211ms/step - loss: 0.0834 - accuracy: 0.970
5 - val_loss: 0.0159 - val_accuracy: 0.9956
Epoch 24/25
66/66 [==============================] - 14s 218ms/step - loss: 0.0452 - accuracy: 0.985
2 - val_loss: 0.0042 - val_accuracy: 0.9989
Epoch 25/25
66/66 [==============================] - 14s 212ms/step - loss: 0.0759 - accuracy: 0.976
7 - val_loss: 0.0034 - val_accuracy: 1.0000
```

- **OUTPUT VALIDATION**
- **DESIGN OF TEST CASES**

Some criteria must be used to evaluate the model's quality and predictability of the response. I have used accuracy will as a metric. The accuracy will be assessed as follows after training.

$$acc(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1(\hat{y}_i = y_i)$$

To validate our trained model, we have designed our test dataset with classes as class0, class 1, class 2. where class 0 refers to 60kmph, class 1 refers to 30kmph, class 2refers to 50kmph.

```
In [4]:
# Resizing the images to 30x30x3
IMG_HEIGHT = 30
IMG_WIDTH = 30
channels = 3

NUM_CATEGORIES = len(os.listdir(train_path))


classes = { 0:'Speed limit (60km/h)',
            1:'Speed limit (30km/h)',
            2:'Speed limit (50km/h)'}

folders = os.listdir(train_path)

train_number = []
class_num = []

for folder in folders:
    train_files = os.listdir(train_path + '/' + folder)
    train_number.append(len(train_files))
    class_num.append(classes[int(folder)])
```

As our model is trained perfectly it will be able to predict the actual outcomes that are desired out of all. It is clearly evident that in all the cases of the testing data the predictedoutput is same as the actual output.The Fig 5.1.a pictures the actual and predictedresults for the testing dataset

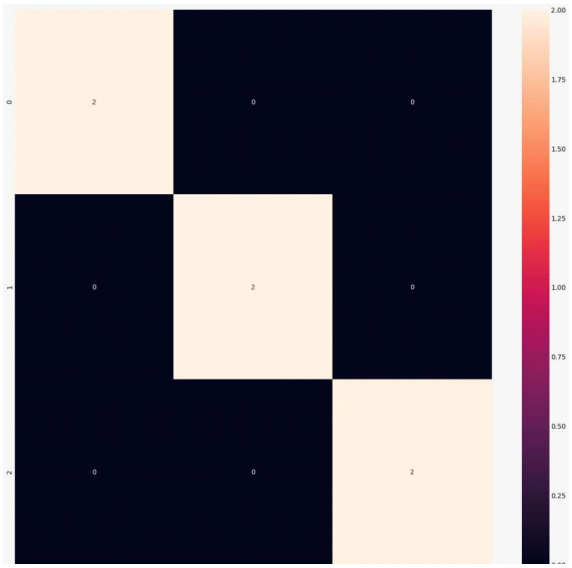


Fig:5.1.b confusion matrix

The Fig 5.1.b is the confusion matrix that describes the predictive analysis

Confusion matrices are used to visualize important predictive analytics like recall, specificity, accuracy, and precision. Confusion matrices are useful because they give direct comparisons of values like True Positives,

False Positives, True Negatives and False Negatives.

From our confusion matrix, we can calculate five different metrics measuring the validity of our model.

- Accuracy (all correct / all) = TP + TN / TP + TN + FP + FN
- Misclassification (all incorrect / all) = FP + FN / TP + TN + FP + FN
- Precision (true positives / predicted positives) = TP / TP + FP
- Sensitivity aka Recall (true positives / all actual positives) = TP / TP + FN
- Specificity (true negatives / all actual negatives) =TN / TN + FP

The model outperformed other benchmark models by giving us accuracy close to 99 percent, which was higher than all other benchmark models. Some criteria must be used to evaluate the model's quality and predictability of the response. I have used accuracy will as a metric. The accuracy will beassessed as follows after training.

- **VALIDATION**

Many different techniques have been applied to detect traffic signs. Most of these techniques are based on using HOG and SIFT features. Several methods for recognizing traffic signs have been published. Moutarde et al. present a neural network-based system for recognizing speed limit signs in Europe

and the United States . They do not, however, provide individual classification results. On average, the overall system, including detection and tracking, achieves a performance of 89 percent for US speed limits and 90 percent for European speed limits. A number-based speed limit classifier is trainedon 2,880 images in [2].

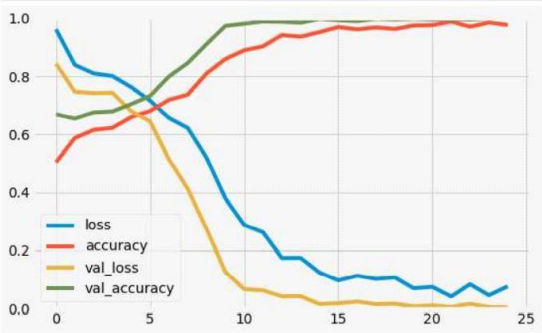
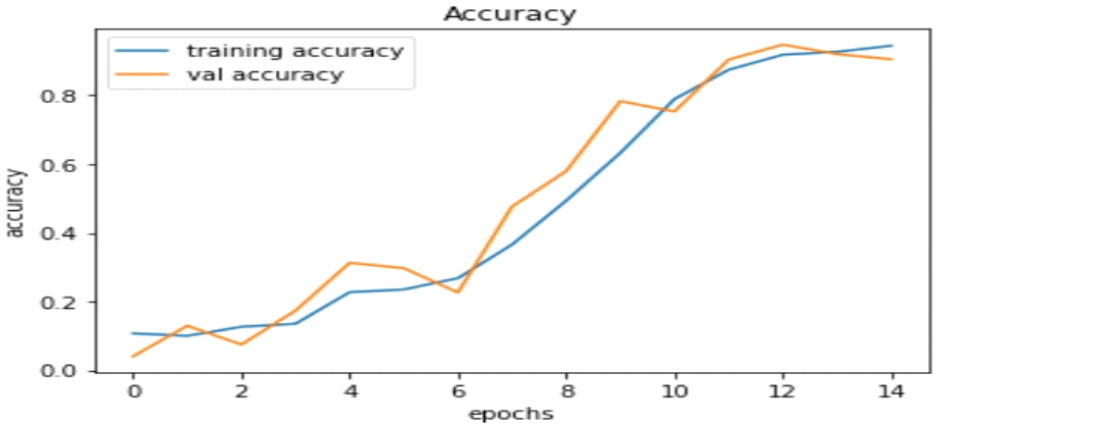
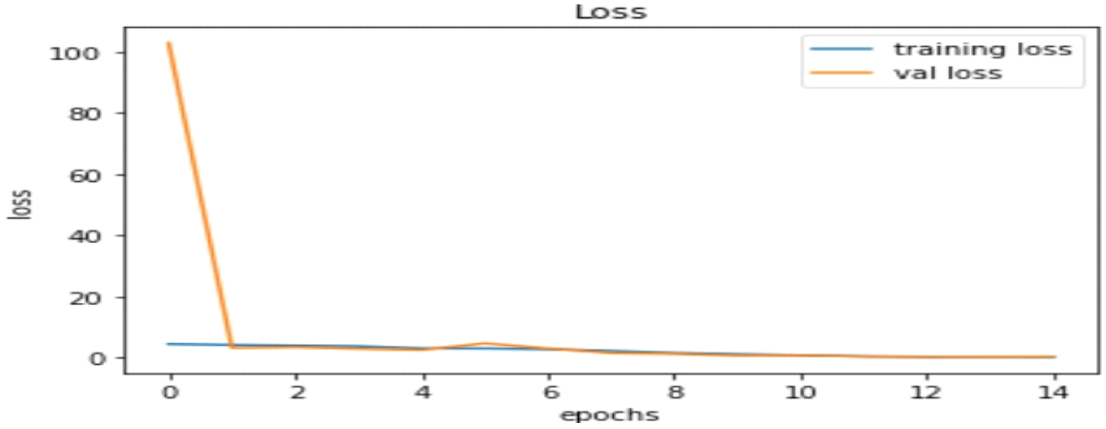


Fig:5.2 validation graph

The Fig 5.2 is a graph that shows thevalidation of the results obtained

It has a 92.4 percent correct classification rate on 1,233 images It is unclear, however, whether images of the same traffic sign instance is shared by multiple sets. In my approach I used biologically inspired convolutional neural networks to build a model which 39 can predict the type of trafficsign with better accuracy than already existing models and Feature Extraction is worth to be noted as well.





CONCLUSION

- **SALIENT CONCLUSION**

  The proposed model has shown better accuracy in detecting various Traffic signs. The most difficult aspect was developing the models and conducting adequate training. This was initially due to the large amount of resources required, as well as the fact that the model was frequently too sophisticated or too basic to detect patterns in data. Training, overfit regulation, and convergence were all difficult at first, and it is a never-ending process. However, once a model begins to converge, increasing accuracy and decreasing loss, it is simply a matter of time and fine tuning to improve it. Of course, we can always make changes to the model. we'll have to figure out when it's good enough for our needs because there isn't a clear end point. Deep learning approach significantly reduces the time cost of training negative samples, ensures the balance of positive and negative samples, and improves the accuracy of the Softmax classifier.

- **SCOPE FOR FUTURE**

  There are a few things that might be done to improve the project. One possible improvement is to improve model further by using Data Augmentation and image prepossessing to improve model result quality. Another improvement is to learn how to identify the text portion of traffic signs and use that information to build a real-time traffic sign system. Another goal for the future is to be able to apply this to self- driving cars.

REFERENCES

- [1]. Base Paper: M. M. William et al., "Traffic Signs Detection and Recognition System using Deep Learning," 2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS), 2019, pp. 160-166, doi:10.1109/ICICIS46948.2019.9014763.
- [2]. Evaluation of Deep Neural Networks for traffic sign detection systems . AlvaroArcos-Garc´ıa∗, Juan A.´Alvarez-Garc´ıa, Luis M. Soria- MorilloDpto.de Languages y Sistema's Inform´attics, Universidad de Sevilla, 41012, Sevilla, Spain *(1)* (PDF) Evaluation of Deep Neural Networks for traffic sign detection systems. Available

- [3]. Miguel Lopez-Montiel, Ulises Orozco-Rosas, Moises Sánchez-Adame, Kenia Pico's, Oscar Humberto Montiel Ross, "Evaluation Method of Deep Learning- Based Embedded Systems for Traffic Sign Detection", Access IEEE, vol. 9, pp. 101217-101238, 2021.