Improved Deep Learning Performance for Real-Time Traffic Sign Detection and Recognition Applicable to Intelligent Transportation Systems

Anass BARODI[1]*, Abderrahim Bajit[2], Abdelkarim ZEMMOURI[3], Mohammed Benbrahim[4], Ahmed Tamtaoui[5]
Laboratory of Advanced Systems Engineering (ISA), National School of Applied Sciences
Ibn Tofail University, Kenitra, 14000, Morocco[1, 2, 3, 4]
National Institute of Posts and Telecommunications (INPT-Rabat)
SC Department, Mohammed V University, Rabat, 10000, Morocco[5]

***Abstract*—In this paper, we improve the performance of Deep Learning (DL) by creating a robust and efficient Convolutional Neural Network (CNN) model. This CNN model will be subjected to detecting and recognizing traffic signs in real-time. We apply several techniques; the first is pre-processing, which includes conversion of color space RGB, then equalization and normalization histogram of the image dataset according to Computer Vision (CV) tools. The second is devoted to Artificial Intelligence (AI), which needs the right choice of a neural layer such convolution layer, or dropout layer, with powerful optimizer as Adam and activation functions such as ReLU and SoftMax. Also, we use the technique of augmentation dataset which characterizes by the function of batch size for each epoch. The results obtained is very satisfactory, the prediction at the average is equal to 98%, which encourages this approach to the integration in Intelligent Transportation Systems (ITS) in the automotive sector.**

Keywords*—Deep learning; convolutional neural network; computer vision; artificial intelligence; traffic sign detection; traffic sign recognition; intelligent transportation systems

I. Introduction

The detection and recognition of traffic road signs are done in different ways, depending on the methodology or strategy followed by the researcher. In general, the detection and recognition methods can be summarized in three classes. The first method can be based on color segmentation (red, blue, yellow) [1]. In the second method, we can use the geometry of objects (Triangular, Square, Rectangle)[2]. Finally, methods that use artificial intelligence (AI), specifically DL of CNN architecture [3]. For road safety, we use ITS systems [4]. This system is devoted to detecting and recognizing all traffic road signs by identifying them from other objects that existed in environments (a passage, animals, cars, trucks, buildings……) in real-time[5]. These systems are used in Advanced Driving Assistance Systems (ADAS) [6][7] and are based on a digital camera for perception road environment.

There is a standard technique for detecting and recognizing traffic road signs. For example, the scale-invariant feature transform (SIFT) [8][9], the local binary patterns (LBP) [10], and the histogram of oriented gradients (HOG) [11]. Also, we find advanced techniques to classify a different object, in which the feature vectors are extracted normally from the training dataset, for example, the support vector machine SVM [12], VGG16 [13], and ImageNet [14]. In recent years, we have been using the CNN model for complex classification situations [15]. The CNN architectures are the best models; they have the same analysis vision as the human being. [16].

To guarantee a reliable and effective model in the decision, most of the research work in the field of AI often plays on the following parameters: optimizer [17], accuracy function [18], loss function [19], dataset [20], architectures [21].

In this paper, we play with several parameters to obtain a robust and efficient model for traffic sign detection and recognition. The first thing we will examine is the effect of normalizing and equalizing the images in the traffic sign dataset on model training. So according to the result of the first step, the second step is choosing an optimal fitting function (Simple, Generator) for deploying the best function between them. Finally, we will use the data augmentation technique by discussing the effect of batch size function during model training. All this is to ensure that the proposed ITS system detects and recognizes signs well in advance so the right decisions can be made as quickly as possible.

In our work, we will test our approach based on Computer Vision (CV) and Artificial Intelligence (AI), for the detection and recognition of the different traffic road signs in real-time. The approach results can be exploited by Intelligent Transport System (ITS) to assist the driver. The paper is organized as follows: Section 1 introduces the most techniques used for the detection and recognition of road signs. Section 2 is dedicated to related work, and then Section 3 presents a general view of the approach proposed. Section 4 is for methodology. Section 5 is devoted to experimentation and evaluation of the approach proposed. Section 6 with Section 7, is for the real-time implementation to recognize traffic road signs. The last section is devoted to the conclusion.

II. Related Work

Most of the developed applications that have high accuracy in object detection and recognition are based on the RNN and CNN architecture [22]. Nevertheless, depending on the available data or the problem to be solved, one type of neural network may be more suitable and used than another for a different problem than the one it is used. Generally, a

Recurrent Neural Network (RNN) is used for text processing and speech recognition as illustrated in Table I. In this regard, convolution networks are applicable for object recognition in images and can specifically identify the shape of objects as illustrated in Table II. In this work, we will use the CNN architecture which is the most efficient neural network model concerning the available dataset.

A. The Constraints of the Traffic Sign Detection and Recognition Algorithms

Detection and recognition based on color segmentation are ranked as one of the fastest methods [41], applied for example to the recognition of road lanes, traffic signs, and vehicle license plates. Most algorithms use this technique to extract regions of interest, by setting specific filters to recognize apparent objects [42]. But this method can meet several problems such as weather conditions (snow, rain...), time of day (morning, night...) which has a great effect on the appearance (light reflection on the signs), or object distance (between the camera and road sign), lead to a false object detection recognition.

Some authors apply a more reliable method, it is the detection and recognition of the geometry of the road signs [43], that the detection is made on the basis of the objects contours in the image. To avoid any overlapping with the objects existing in the road environment by a structural analysis of the road signs [44].

TABLE I. MOST RNN ARCHITECTURE APPLICATIONS

Applications	RNN Architecture	Reference
Text processing	Efficient RNN Text Classification	J. Du [23] H.Chen [24] Z. Parcheta [25]
	Medical Text Classification Framework	X. Li [26] M. Ibrahim [27]
Speech recognition	Anticipation-RNN to Interactive Music Generation	F. Nielsen [28] D. Bisharad [29]
	Sentiment Analysis	A. Onan [30] J. Huan [31]

TABLE II. MOST CNN ARCHITECTURE APPLICATIONS

Applications	CNN Architecture	Reference
Image recognition	Traffic sign recognition systems	Á. Arcos-García [32] Á. Arcos-García [33]
	Lane Detection in Traffic Scene	J. Li [34] J. Kim [35] J. Tang [36]
Form recognition	CNN Design for Real-Time Traffic Sign Recognition	A. Shustanov [37] F. Shao [38]
	CNN Network for Real-life Traffic Sign Detection	T. Yang [39] Á. Arcos-García [40]

B. Deep Learning and Neural Network

The learning methods are among the techniques that use DL [45], this method has made a revolution in the industrial sector, especially in the embedded systems in the automotive sector [46]. This method is robust in object detection and recognition compared to the geometric and colorimetric methods, which are among the classic methods that suffer from many factors.

The creation of CNN models was based on neural networks. Many hidden layers of the neural network serve to produce CNN. These neuron layers are grouped into a tree category of layers, input layers, hidden layers, and output layers. Firstly, the feature vectors dataset is accepted from the input layer and has a bias neuron. Secondly, the liaison between input and output is hidden layers that use the neuron bias. Finally, the output of neural networks is not used for the bias neuron. The output from a single neuron is calculated according to the following equation (1).

$$f(x,\theta) = \phi(\sum_i(\theta_i . x_i) \quad (1)$$

- The input vector (x) represents the feature vector.
- The vector θ represents the weights.
- The ϕ is the transfer/activation function.

III. THE APPROACH DESCRIPTION

The approach that will be proposed to integrate it into an ITS system, is essentially based on the creation of a CNN architecture to guarantee road safety for both passengers and drivers of vehicles. Therefore, our approach is based on two processes, the detection process and the recognition of traffic signs as shown in Fig. 1. The detection process uses camera-based CV techniques to receive images to ensure that traffic signs are detected. When the signs are detected, the recognition process is activated using AI techniques. We will use a CNN architecture to extract the characteristics of the road signs. To achieve our objective, the deep training will be done on the German Traffic Sign Recognition Benchmark (GTSRB) dataset. We arrive at the end to identify each detected by the classes that belong to the prediction probability.

The strategy we will follow to have an efficient CNN architecture is summarized in the following points:

- Transformation techniques (equalization and normalization histogram).
- Creation of CNN architecture (convolution layers, and max-pooling layers)
- DL of the CNN architecture with simple fit function and generator fit function.
- Testing the performance of the CNN model in real-time detection and recognition of traffic road signs.

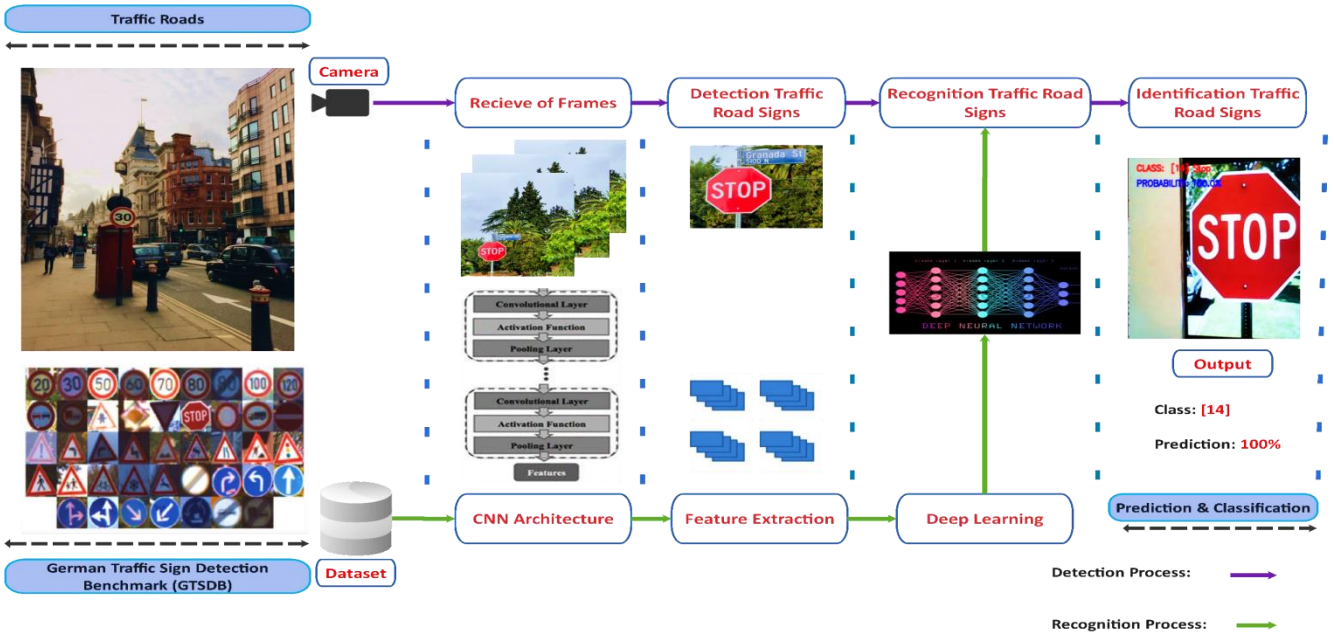


Fig. 1. General View of Approach Applicable for the ITS System.

IV. Methodology

A. Transformation Techniques for Dataset

a) Visualization dataset: For our implementation, we use a dataset of the German traffic sign Benchmark [47], composed tree part, training data, validation data, and testing data. The training set uses 80% of the data and the validation set uses 20%. The GTSRB is composed of 43 traffic road sign classes, 34799 images for training data, 4410 images for validation data, and 12630 images for testing data, as illustrated in Fig. 2.

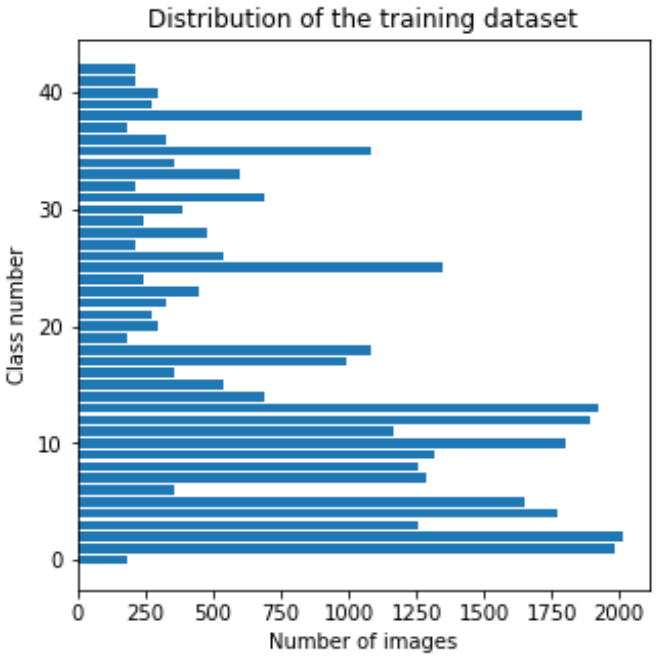


Fig. 2. Visualization of GTSRB Training Datasets.

b) Normalization of a histogram: Normalizing a histogram is a technique consisting of transforming the discrete distribution of intensities into a discrete distribution of probabilities [48]. To do this, we need to divide each value of the histogram by the number of pixels. In our case, the normalization is done by dividing all pixels in an image by 255.

c) Equalization of a histogram: Histogram equalization is an image processing method to adjust the contrast of an image, by modifying the intensity distribution of the histogram [49]. Equalization processing is based on the use of the cumulative probability function. This function is a cumulative sum of all the probabilities in its domain and is defined by equation (2).

$$cdf(x) = \sum_{k=-\infty}^{x} P(k) \quad (2)$$

The idea of this processing is to give the resulting image a linear cumulative distribution function.

B. Convolutional Neural Networks (CNNs)

Domain CV has been affected by AI mainly by CNNs. The neural network architecture was introduced by LeNet-5 [50]. The next step is the description of each layer type used in the CNN model.

a) Convolution layers: The first layer of analysis is the convolution, it allows us to detect the characteristics of each visual element: circles, lines, colors, edges ..., this work is done by internal filters in the layer. If the number of filters is very well brought, they have more features for better accuracy. The filters have a square shape that sweeps over the image from the right to the left. Then there is a very important parameter, which is the width and length of the filter that normally affects the number of features extracted from the images. The single

output matrix of the convolution layer is described in equation (3).

$$M_j = g\left(\sum_{i=1}^{N} Img_i * Ker_{i,j} + b_j\right) \quad \# \quad (3)$$

I_{mg} : Input matrix. Ker: Kernel matrix.

bj: Bias. g: Non-linear activation.

Each set of kernel matrices represents a local feature extractor that extracts regional features from the input matrices. Optimizes neural network connection weights, and can be applied here to train the kernel matrices, biases as shared neuron connection weights.

b) Max pooling layers and dropout layers: Putting the Max-Pooling layers belong after every convolution layer. It serves for re-sizing a picture of 2D in a smaller dimension [51]. Most CNN frameworks implement dropout as a separate layer to avoid the production in DL the overfitting. Dropout layers function like a regular, densely connected CNN layer. The only difference is that the dropout layers will periodically drop some of their neurons during training.

c) Activation function: However, current deep neural networks mainly use the following activation functions, each function has a role to play in a neural network. For the output of the hidden layers, we use the ReLU (Rectified Linear Unit) function [52]. The ReLU function is calculated as follows in equation (4).

$$\phi(x) = max\ (0, x) \quad \# \quad (4)$$

The ReLU activation function [53][54] was one of the key improvements in CNN applications, that make deep learning. Unfortunately, the ReLU function is not differentiable at the origin, which makes it hard to use with backpropagation training. ReLU for rectified the feature map, to find the final value positive and deleted the negative value.

The output of classification CNN: We implemented SoftMax. The SoftMax is calculated as follows in equation (5).

$$\phi_i(z) = \frac{e^{z_i}}{\sum_{j \in group} e^{z_j}} \quad \# \quad (5)$$

The SoftMax function is only useful with more than one output neuron. It guarantees that the sum of all output neurons is equal to 1.0. It is therefore very useful for classification, where it indicates the probability that each of the classes is the correct choice.

d) Optimization function: Adam optimizer is very effective [55]. Adam estimates the first mean and second variance moments to determine the weight corrections. Adam begins with an exponentially decaying average of past gradients (m) described in equation (6).

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad \# \quad (6)$$

g_t : the gradient at time t.

This average accomplishes a similar goal as a classic momentum update; however, its value is calculated automatically based on the current gradient (g_t). The update rule then calculates the second moment (v_t) in equation (7).

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) {g_t}^2 \quad \# \quad (7)$$

The values m_t and v_t are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients respectively. β_1 and β_2: are exponential decay rates. Adam is very tolerant of the initial learning rate (η) and other training parameters. Default values of β_1=0.9, β_2=0.999, and $\eta=10^{-8}$ [45].

C. CNN Architecture

We have a dataset of dimensions (32,32,3), and we will perform a conversion from RGB color space to gray level. The input images of our architecture will have dimensions (32,32,1). Table III presented the architecture of CNN in detail, type of layers, output shapes, and activation functions. The layers with their corresponding type are shown, denoting the characteristics used. Then implementation of CNN in CPU takes more time because we have a dataset of images that are more difficult to execute. However, the faster implementation we propose to use GPU.

TABLE III. PROPOSED CNN ARCHITECTURE

Layer	Type	Output shape	param	Activation
Conv2d_1	Conv2D	(None, 28, 28, 60)	1560	ReLU
Conv2d_2	Conv2D	(None, 24, 24, 60)	90060	ReLU
Max_pooling2d_1	Max_pooling2d	(None, 12, 12, 60)	0	N/A
Conv2d_3	Conv2D	(None, 10, 10, 30)	16320	ReLU
Conv2d_4	Conv2D	(None, 8, 8, 30)	8130	ReLU
Max_pooling2d_2	Max_pooling2d	(None, 4, 4, 30)	0	N/A
Dropout_1	Dropout	(None, 4, 4, 30)	0	N/A
Flatten_1	Flatten	(None, 480)	240500	N/A
Dense_1	Dense	(None, 500)	0	ReLU
Dropout_2	Dropout	(None, 500)	21543	N/A
Dense_2	Dense	(None, 43)		Softmax

Total params: 378,023
Trainable params: 378,023
NON TRAINABLE PARAMS: 0

V. EXPERIMENTATION AND EVALUATION

The results are implemented in ASUSTek Computer, processor intel® Core™ i7-7500 CPU @2.70GHz 2.90GHz, Memory installed (RAM): 8,00 Go, exploitation System 64 bits, processor 64 bits Systems Model: X541UJ, GPU NVIDIA GeForce 920M using the TensorFlow, Keras, and OpenCV libraries.

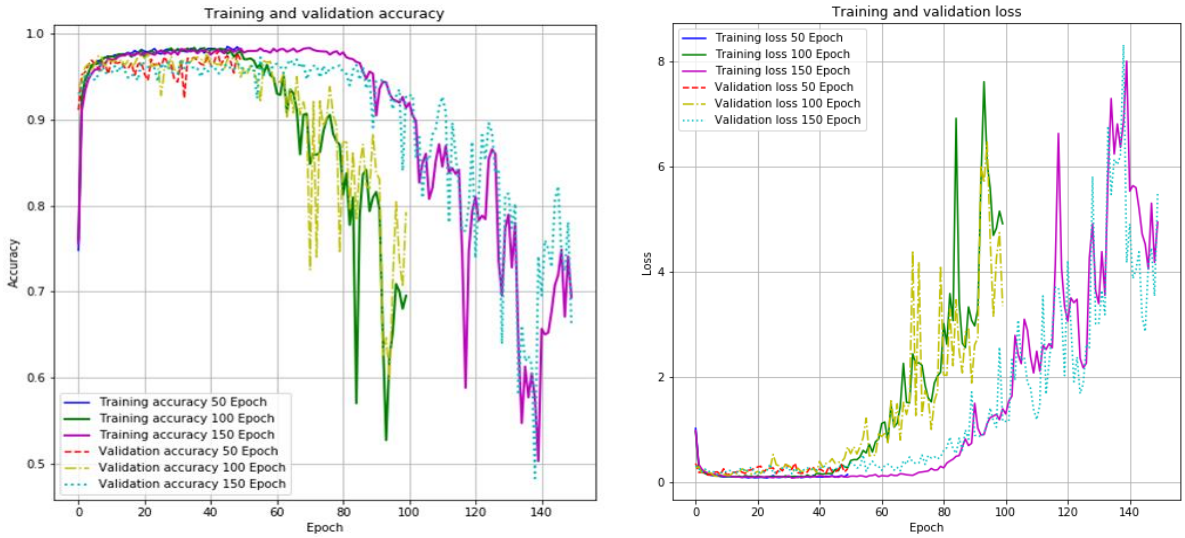
A. Simple Fit Function

The training of the proposed CNN model requires two essential elements, the training data, and the training labels. For the training, we will use the fit function of the Keras library.

The number of epochs is the number of times the model will run through the data. The more epochs we run, the more the model will improve, up to a certain point. We started our model for 50 epochs with a batch size set to 32. We will also train the dataset with equalization and normalization of the histogram. Thus, the training without equalization and normalization will be noted as Method 1, and the training with equalization and normalization will be noted as Method 2.

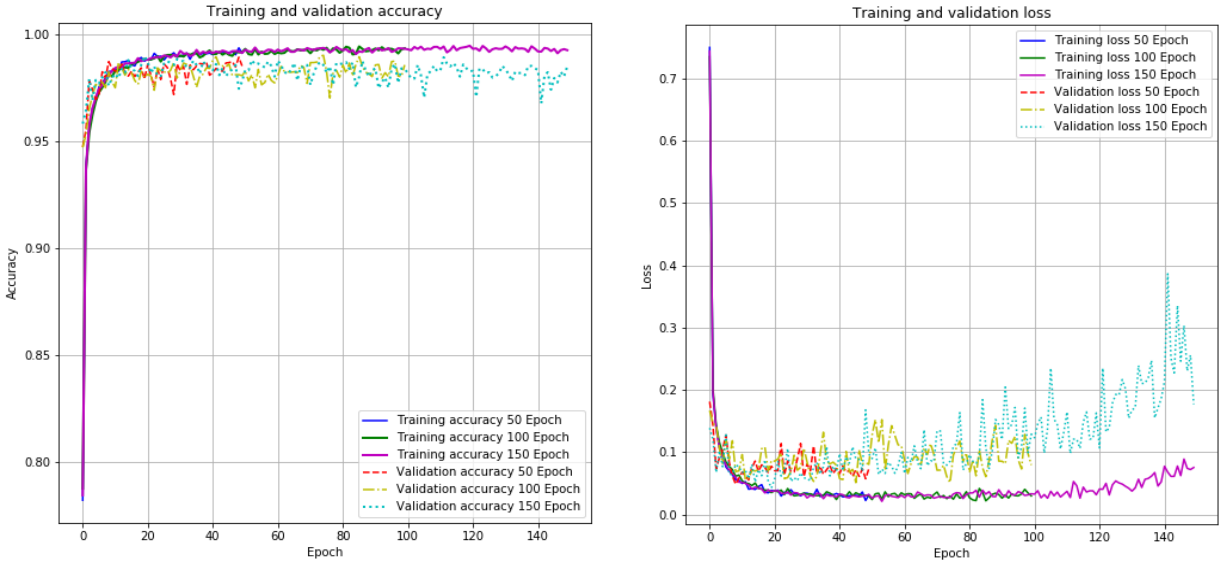
We can visualize in Fig. 3 in the accuracy curve, a drop during the training of the data in 2 steps for 50 and 100 epochs. But for the loss curve, we have a huge increase in the error value. So, method (1) leads us to overfit. We can deduct from Fig. 4 that we don't have any underfitting or overfitting in the accuracy curve, we can easily observe that the increase in the number of epochs did not disturb the learning stability. The same thing for the loss curve, we have a very remarkable degradation of the error values compared to the curve of method (1). Equalization and normalization can be used almost. However, this method (2) shows negligible effect loss and we have the full precision of our network that shows a significant improvement.

A comparison of the performance in Table IV sows accuracy function and loss function. We can conclude from Table IV which contains tests accuracy and loss for Method (1) and Method (2). It is necessary to equalize and normalize. The equalization is served to adjust the contrast in the image's dataset. For the normalization, it allows making training faster and the loss becomes more circular symmetric. The next step is to change the simple fit function by using a fit generator, we visualize if does good predictions and evolution of accuracy with a loss function. The equalization and normalization algorithms result in improved performance of CNN classification.



(a) (b)

Fig. 3. Method (1): (a) Training and Validation Accuracy, (b) Training and Validation Loss.



(a) (b)

Fig. 4. Method (2): (a) Training and Validation Accuracy, (b) Training and Validation Loss.

TABLE IV. PERFORMANCE COMPARISON OF METHOD 1 AND METHOD 2

Method		Model	Fit Function	Learning Rate	Loss Function	Optimizer	Train Dataset	Epochs	Test Accuracy (%)	Test Loss (%)
Method 1	Without Equalization and Normalization images Datasets	CNN	Simple	10^{-2}	Categorical Cross-Entropy	Adam	34799	50	94.63	44.11
								100	78.21	35.09
								150	64.18	5.76(>100%)
Method 2	With Equalization and Normalization images Datasets							50	96.43	17.65
								100	96.19	25.48
								150	96.52	42.29

B. Fit Generator Function

We propose to use the fit generator function to accept the data sets, perform backpropagation, and update the weights in our model. This function has a hyperparameter, it is the number of steps per epoch, its value as the set of servant landmarks becomes divided by the batch size. It is based on an infinite loop, which must not return empty or exit. However, all researchers calculate the value of steps per epoch as the total number of training data divided by the batch size of training data images.

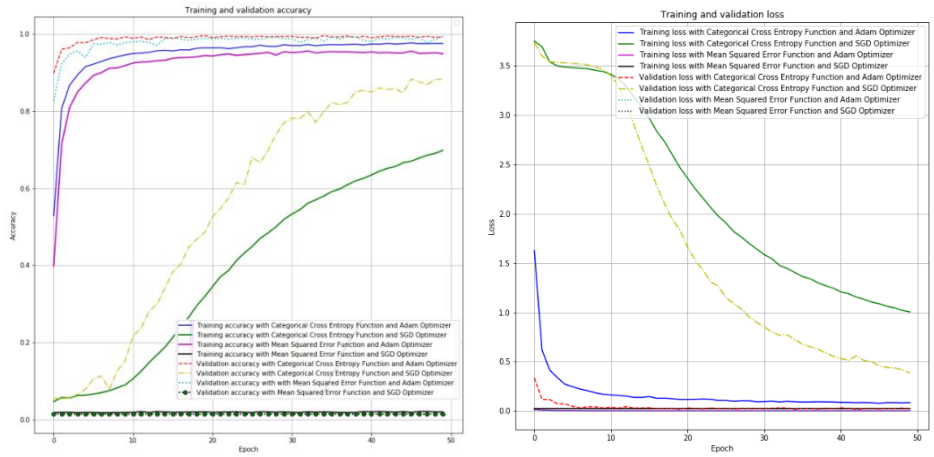
So, the idea of our experiment is to use method 2 from the previous section. Method 2 will be driven by the generator fitting function with a batch size of 32. We will compare different optimizers (Adam and SGD) and the loss function (Categorical cross-entropy, and Mean squared error). We fixed parameters learning rate in 10^{-2} and epochs in 50.

In Table V, when the loss function uses categorical cross-entropy, we have a high prediction score with a low loss score. Now we improved the model to get the lowest loss score. We got the best scores with the Adam optimizer and the categorical cross-entropy function, for 97.11% accuracy and 11.32% loss. Moreover, the idea is now to improve the accuracy score.

As we can see in Fig. 5, using the fit generator function in the training model the objective is achieved, at 90% we control the situations for not have the overtraining our DL models. The assumptions are therefore correct, we using all of the datasets at each epoch. We need to choose a batch size and steps per epoch which multiply to give a total number of samples. Usually, it will be a resource. If memory is a problem, we need to reduce the batch size until we can adapt a batch on a GPU. Note that this implementation also allows us to use the multiprocessing argument of a fit generator, where the number of threads specified in workers corresponds to those which generate batches in parallel. A fairly high number of workers ensure that the calculations performed on the GPU are managed efficiently, or in other words, the bottleneck of the whole training process will be due to the propagation operations. In our case, we would probably set batch size the desired amount; we change it only if you want the model to not use all the data for each epoch which deflects the definition of the word epoch.

TABLE V. EXPERIMENT RESULTS OF OPTIMIZER AND LOSS FUNCTION

Loss function	Model	Fit Function	Optimizer	Test Accuracy (%)	Test Loss (%)
Categorical Cross-Entropy	CNN	Generator	SGD	86.74	46.27
			Adam	97.11	11.32
Mean Squared Error			SGD	01.16	02.27
			Adam	97.08	12.12



(a) (b)

Fig. 5. (a) Training and Validation Accuracy, (b) Training and Validation Loss.

VI. Discussion

We introduce one more technique to improve the model training process data augmentation. This technique creates new data for our CNN model to use during the training process. This is done by taking our existing datasets and transforming or altering the images in useful ways to create new images.

A. Image Data Generator Function

We can have a typical sign image such as this STOP sign image, taking this image and transforming it to create a different image representing the same stop sign. The transformation could be rotation or simply zooming into the image. Also, could even be a combination of both these transformations. These newly created images are referred to as augmented images because they essentially allow us to augment our dataset by adding them. The data augmentation technique is useful because it allows our model to look at each image in our dataset from a variety of different perspectives. This allows it to extract relevant features more accurately and allows it to attain more feature-related data from each training image. This is especially the case for our traffic sign datasets because we have a small dataset (32x32) and a large number of classes. This means that certain classes have very few proximately only 200 training in the Fig. 2. It can benefit our traffic sign recognition model.

We apply the five following transformations with shift range, height shift range, zoom range share, and a rotation range. Five transformations will add sufficient variety to GTSRB datasets and will allow the training process to be much more effective. The first transformation is with shifts, this refers to a horizontal translation in the image which will cause our images to be centered, and this will help our CNN model adapt to test images that aren't necessarily going to be centered. The range can be defined in two ways, if the range value is defined as a number between 0 and 1, then it refers to the fraction of the image that can be shifted. A value of 0.1 would simply imply that the maximum horizontal shift possible is 10 percent of the width of the image. The images with only horizontal translation can be similar. So, to have a difference between the generated, we apply a second technique is a vertical translation. The range value is defined in much the same way and for that reason; the value of vertical translation is 0.1 (10%).

For zoom transformation, can be either zoom out or into the image. The degree of zoom can be defined with a float value between 0 and 1. While the maximum outer zoom is defined by one minus the float value and the maximum zoom is defined by a 1 plus the flow value. We will use a float value of 0.1 which means that we can zoom as far as 0.1 eight's and zoom in as close as 0.2. Next, we have the shear transformation in plane geometry a shear mapping is a linear map that displaces, each point in a fixed direction by an amount proportional to its side and distance. The line that is parallel to that direction, possible in both the x-direction and the y-direction. This transformation is defined using shear intensity which simply refers to the magnitude of the shear, angle in degrees as seen in the image above. We apply a small magnitude of shear to be effective, using a value of 0.1. The last transformation is the rotation; this transformation is a bit more intuitive it simply rotates an image by a certain value of degrees. This value can be defined using an integer value, in our case, we will use 10. These transformations are simply applied to stop signs as shown in Fig. 6, which will then be applied to the GTSRB dataset.

B. Batch Size Function

First, we declare a batch size is equal to 32 which mean that our image generator will create a batch of 32 images at a time for our CNN model to use our next argument as illustrated in Fig. 7. Also, the steps per epoch this parameter essentially refers to the number of batches. The steps per-epoch argument must specify the number of batches of samples comprising one epoch. In our case, the original dataset has 34799 images and the batch size is 32. Then a reasonable value for steps per epoch when fitting a model on the augmented data might be ceil (34799/32), or 1087 batches. So, we fix the value of the steps per epoch in 1000.

C. Experimental Results

We are fixed step pre-epochs to 1000, we switch the value of epochs between 50 and 150, we behold augmentation accuracy the same time value loss has diminution. We fit and evaluate all these models in different batch sizes (32, 64, 128, and 256) using the same procedure above of optimizer Adam and the same value of steps pre-epochs with different epochs, found through some minor experimentation. The model is evaluated, reporting the classification accuracy on the test sets between 96.86% and 98.01%. We can specify the results may vary given the stochastic nature of the training algorithm. Table VI demonstrates the effect of batch size, after testing very hard which took an enormous time to find it up to incredible values. When we have for batch size is 256, we have a precision in 50 epochs of 98,01% which is very interesting, and also a remarkable reduction in the function error of 09.15%. The same thing for size 100 epochs has values for the two 97.99% and 09.11%.

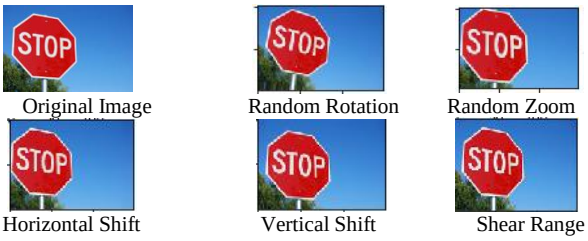


Fig. 6. Different Transformation for Dataset Augmentation.

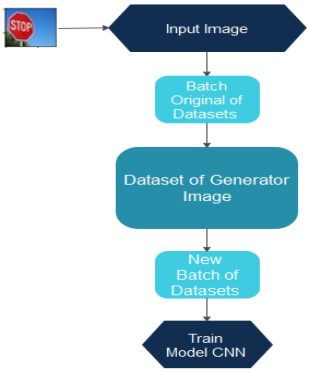


Fig. 7. The Batch of the Training Dataset GTSRB.

TABLE VI. FLOW DATA FOR BATCH SIZE FUNCTION

Batch Size	Optimizer	Step pre-epochs	Epoch	Test Accuracy (%)	Test Loss (%)
32	**Adam**	1000	50	**97.15**	**10.94**
			100	**97.11**	**11.32**
			150	**96.86**	**14.16**
64	**Adam**	1000	50	**97.18**	**11.90**
			100	**97.04**	**12.28**
			150	**97.69**	**09.25**
128	**Adam**	1000	50	**97.38**	**11.02**
			100	**97.85**	**10.74**
			150	**97.94**	**09.68**
256	**Adam**	1000	50	**98.01**	**09.15**
			100	**97.99**	**09.11**
			150	**97.83**	**10.84**

In Fig. 8, we can see the validation converges to above 99%. A significant improvement is shown over our previous CNN model. This might be our modification that was pretty effective. We have a much smaller gap and training accuracy as well as our validation loss and accuracy, respectively. This demonstrates consistency in our training and a better-trained model and we now finish our model training with a validation accuracy of over 98 % and training accuracy. This is all very good to see and shows our augmentation technique was effective. The model will not learn complex patterns and we can avoid overfitting, we use more dropout layers in our architecture and check its performance. So, the augmentation dataset after performing histogram normalization and equalization, the model learned the data better, and the accuracy of the set improved. Now there is just one more test that our model needs to pass and that is classifying images from the test dataset to predict a couple of them correctly. So, we'll start by testing out the image not seen before for our CNN model.

D. Analyses Performance of a Model Trained

We define several measures based on the confusion matrix, to quantify the performance of a classifier from different points of view: Precision by class, average precision, Recall by class, average recall, F-score by class, and f-score average.

a) Precision of classification: The accuracy of a classifier concerning a certain class in other words, about a certain modality of the variable to be predicted, is measured as the proportion of individuals, among all those for whom the classifier predicted this class, who belong to it, exposed in this equation:

$$p_i = \frac{TP}{TP+FP} \quad \# \quad (8)$$

TP :(True Positive) Element of the class correctly predicted.

FP: (False Positive) Element of the class badly predicted.

The overall means of the precision over all the classes i can be evaluated by the macro-average which first calculates the precision on each class i followed by a calculation of the average of the details on the n classes based on this equation:

$$Precison = \sum_i^n \frac{p_i}{n} \quad \# \quad (9)$$

p_i: precision each class i. n : number of classes.

b) Recall of classification: The recall of a classifier with a certain class is measured, like the proportion of individuals, among all those who belong to this class, for which the classifier predicted this class.

$$r_i = \frac{TP}{TP+FN} \quad \# \quad (10)$$

The global averages of the recall over all of the classes i can be evaluated by the macro-average which first calculates the recall over each class i followed by a calculation of the average of the reminders over the n classes:

$$Recall = \sum_i^n \frac{r_i}{n} \quad \# \quad (11)$$

r_i: recall each class i. n : number of classes.

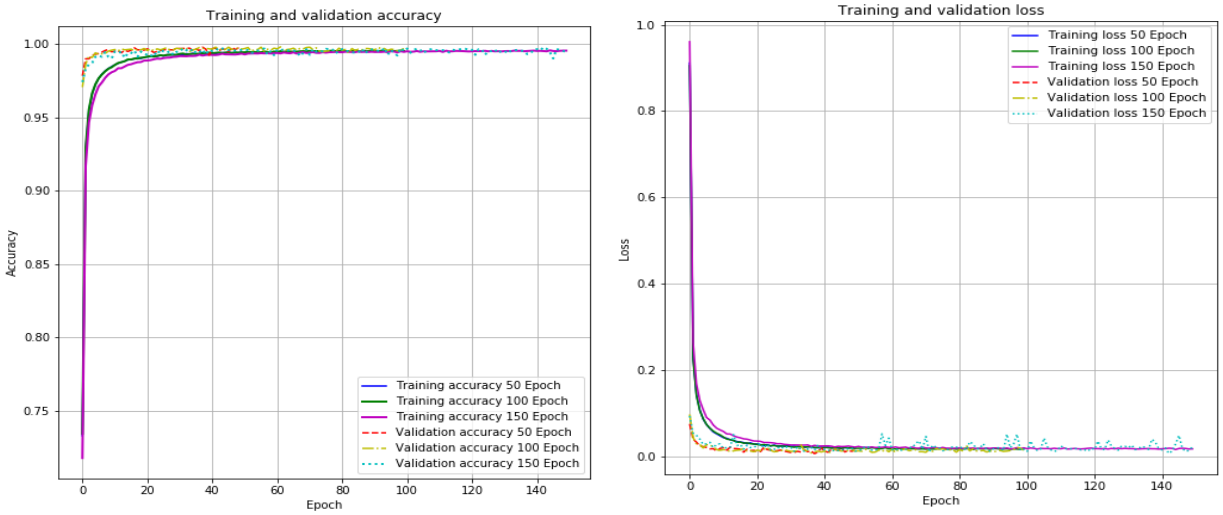


Fig. 8. Accuracy and Loss Curves of Epochs with a Batch Size of 256.

c) F-score of classification: We can summarize the recall precision measurements to a class in a single indicator, by calculating the harmonic mean:

$$R\,F-score_i = \frac{2*(p_i*r_i)}{p_i+r_i} \qquad \# \qquad (12)$$

p_i: precision each class i. r_i: precision each class i.

The average over each class of these indicators gives global indicators on the quality of the classifier.

$$F_{-Score} = \frac{2*(Precision*Recall)}{Precision+Recall} \qquad \# \qquad (13)$$

$Precision$: The average precision of all classes.

$Recall$: The average recall of all classes.

E. Confusion Matrix

The Confusion Matrix identifies the classes of signs and also gives the number of times it gets the confused class to identify the class from another in Fig. 9. Most of the color is diagonal, but there are still some annoying spots somewhere. When we narrowly look at the confusion matrix, we see that the classes [0] have very less respectively all classes, but it's minimized for other classes. The diagonal observations are the true positives of each class and other non-diagonal observations are incorrect classifications of the model.

F. Classifier Metrics

A Classification report is used to measure the quality of predictions from a classification algorithm. We can see in the Table VII, the model has the as recall and precision are calculated for individual classes, have a good score of all the class of traffic road signs. We use macro or micro or weighted scores of recalls, precision, and F1 score of a model for multiclass classification problems have a higher score is 98% this very satisfied.

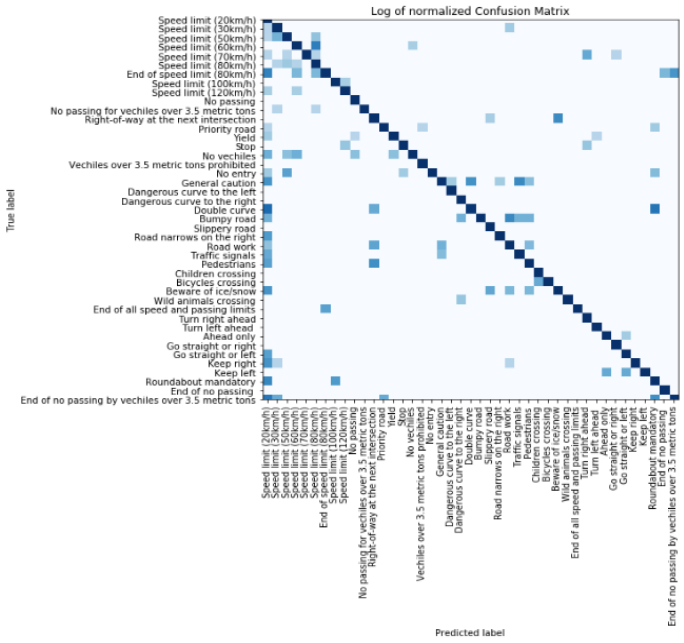


Fig. 9. Confusion Matrix Epochs 100 and Batch Size 256.

TABLE VII. CLASSIFIER REPORT FOR THE CNN MODEL

Class names	precision	recall	f1-score	support
Speed limit (20km/h)	0.98	1.00	0.99	60
Speed limit (30km/h)	0.99	1.00	0.99	720
Speed limit (50km/h)	0.99	0.99	0.99	750
Speed limit (60km/h)	0.99	0.94	0.96	450
Speed limit (70km/h)	1.00	0.98	0.99	660
Speed limit (80km/h)	0.95	0.99	0.97	630
End of speed limit (80km/h)	0.99	0.90	0.94	150
Speed limit (100km/h)	1.00	1.00	1.00	450
Speed limit (120km/h)	1.00	1.00	1.00	450
No passing	1.00	1.00	1.00	480
No passing for vechiles over 3.5 metric tons	1.00	1.00	1.00	660
Right-of-way at the next intersection	0.98	0.96	0.97	420
Priority road	1.00	0.99	1.00	690
Yield	1.00	0.99	1.00	720
Stop	1.00	0.99	0.99	270
No vechiles	1.00	0.97	0.98	210
Vechiles over 3.5 metric tons prohibited	0.99	1.00	1.00	150
No entry	1.00	0.97	0.99	360
General caution	0.99	0.89	0.94	390
Dangerous curve to the left	0.98	1.00	0.99	60
Dangerous curve to the right	0.98	1.00	0.99	90
Double curve	0.85	0.78	0.81	90
Bumpy road	1.00	0.93	0.96	120
Slippery road	0.98	1.00	0.99	150
Road narrows on the right	0.99	0.98	0.98	90
Road work	0.98	0.97	0.97	480
Traffic signals	0.91	0.98	0.95	180
Pedestrians	0.89	0.95	0.92	60
Children crossing	0.99	1.00	1.00	150
Bicycles crossing	1.00	0.99	0.99	90
Beware of ice/snow	0.89	0.95	0.92	150
Wild animals crossing	1.00	1.00	1.00	270
End of all speed and passing limits	1.00	0.98	0.99	60
Turn right ahead	0.96	1.00	0.98	210
Turn left ahead	0.99	1.00	1.00	120
Ahead only	1.00	1.00	1.00	390
Go straight or right	0.99	1.00	1.00	120
Go straight or left	0.97	0.98	0.98	60
Keep right	1.00	0.97	0.98	690
Keep left	1.00	0.98	0.99	90
Roundabout mandatory	0.85	0.93	0.89	90
End of no passing	0.98	1.00	0.99	60
End of no passing by vechiles over 3.5 metric tons	0.95	0.87	0.91	90
micro avg	0.98	0.98	0.98	12630
macro avg	0.98	0.97	0.97	12630
weighted avg	0.99	0.98	0.98	12630
samples avg	0.98	0.98	0.98	12630

VII. Testing the Model

A. Test with the Test Dataset

A remarkable performance is illustrated in Fig. 10. Now we'll test for test datasets, we look reaction to our model, to see where the fails. We tried to visualize the class predictions of the test images, it is relevant to have good results, all the images were well classified, and the curve shown next to each image represents the class of the images among the 42 classes, when we have the color blue and a single peak in the curve means the image has been put in the right place without any errors.

B. Testing the Proposed CNN Model in Real-Time

In this section, we will present and evaluate the results of our approach. Traffic road signs that appear in video sequences are often detected. More details on the video sequences are given in Table VIII. In general, for all performance indicators, our proposed approach outperforms other object detection algorithms by achieving up to 100% accuracy. Our CNN model metric value is often higher than in the results of previous work. For the video sequences, our algorithm surpasses the good probability of prediction and classes of Traffic Road signs by the method. This shows that using a robust appearance CNN model achieves better results. It can also be observed that the CNN precision value obtained for the video sequence is higher than that obtained by the approach with a difference between 97.56% and 100%.

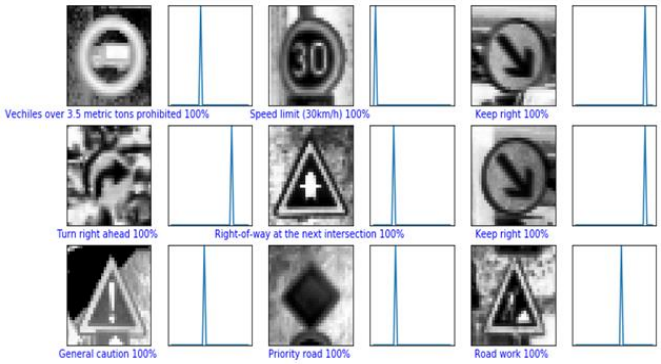


Fig. 10. GRTSB Test Datasets.

TABLE VIII. Test of New Images in Real-Time

Traffic Road Signs Recognition	Classification	Prediction (%)
	Class Number: [17] No Entry	**97.56%**
	Class Number: [12] Priority Road	**99.15%**
	Class Number: [5] Speed limit (80km/h)	**98.4%**
	Class Number: [14] STOP	**100%**
	Class Number: [34] Turn let ahead Class Number: [33] Turn right ahead	**100%**

VIII. CONCLUSION

In this paper, we proposed a methodology for the construction robust CNNs model. We talked about the problems associated with the detection and recognition of traffic road signs in real-time. We also demonstrated how using the right tools and techniques helps us in developing robust CNN models. These CNNs can guarantee road safety in real-time. We also try other pre-processing techniques to further improve the model's accuracy (equalization and normalization histogram). The step of adding augmentation data improved the performance of our deep learning CNN model. We are curious about how much the accuracy can be improved based on adding such simple transformations. We think these results could further be used in the development of automotive systems, such as intelligent transportation systems (ITS). All this is for the safest roads; we try in the future to get better performance and optimist. It is also very interesting to note that the proposed CNN model reaches 98% accuracy using NVIDIA's GPU processor, which makes them feasible for real-time traffic sign recognition.

In future work, we plan to study other neural network architectures that have been shown to be optimistic for traffic sign detection or classification. In addition, we will attempt to employ these networks in advanced in-vehicle platforms applicable to intelligent transportation systems to reveal valuable information that will help drivers make the right decisions in the real world.

REFERENCES

[1] X. Xu, J. Jin, S. Zhang, L. Zhang, S. Pu, and Z. Chen, "Smart data driven traffic sign detection method based on adaptive color threshold and shape symmetry," Futur. Gener. Comput. Syst., vol. 94, pp. 381–391, 2019, doi: 10.1016/j.future.2018.11.027.

[2] S. Kumar, S. D. K, P. Maddula, and N. V. V. Ravipati, "Unified approach for detecting traffic signs and potholes on Indian roads," J. King Saud Univ. - Comput. Inf. Sci., no. xxxx, 2021, doi: 10.1016/j.jksuci.2021.12.006.

[3] D. Sirohi, N. Kumar, and P. Singh, "Convolutional neural networks for 5G-enabled Intelligent Transportation System : A systematic review," Comput. Commun., vol. 153, no. January, pp. 459–498, 2020, doi: 10.1016/j.comcom.2020.01.058.

[4] S. L. G. Eliza, "Embedded real-time speed limit sign recognition using image processing and machine learning techniques," vol. 28, pp. 573–584, 2017, doi: 10.1007/s00521-016-2388-3.

[5] A. Zemmouri, M. Alareqi, R. Elgouri, M. Benbrahim, and L. Hlou, "Integration and implimentation system-on-aprogrammable-chip (SOPC) in FPGA," J. Theor. Appl. Inf. Technol., vol. 76, no. 1, pp. 127–133, 2015.

[6] S. Jagannathan, M. Mody, and M. Mathew, "Optimizing Convolutional Neural Network on DSP," pp. 371–372, 2016.

[7] A. Zemmouri, R. Elgouri, M. Alareqi, M. Benbrahim, and L. Hlou, "Design and implementation of pulse width modulation using hardware/software microblaze soft-core," Int. J. Power Electron. Drive Syst., vol. 8, no. 1, pp. 167–175, 2017, doi: 10.11591/ijpeds.v8i1.pp167-175.

[8] W. Arman, S. Arefin, A. S. M. Shihavuddin, and M. Abul, "DeepThin : A novel lightweight CNN architecture for traffic sign recognition without GPU requirements," Expert Syst. Appl., vol. 168, no. August 2020, p. 114481, 2021, doi: 10.1016/j.eswa.2020.114481.

[9] A. Zemmouri, R. Elgouri, M. Alareqi, H. Dahou, M. Benbrahim, and L. Hlou, "A comparison analysis of PWM circuit with arduino and FPGA," ARPN J. Eng. Appl. Sci., vol. 12, no. 16, pp. 4679–4683, 2017.

[10] Y. Saadna, "Speed limit sign detection and recognition system using SVM and MNIST datasets," Neural Comput. Appl., vol. 0123456789, 2019, doi: 10.1007/s00521-018-03994-w.

[11] A. R. Dubey, N. Shukla, and D. Kumar, "Detection and Classification of Road Signs Using HOG-SVM Method," pp. 49–56.

[12] Z. Liu, M. Qi, C. Shen, Y. Fang, and X. Zhao, "Cascade saccade machine learning network with hierarchical classes for traffic sign detection," Sustain. Cities Soc., vol. 67, no. January, p. 102700, 2021, doi: 10.1016/j.scs.2020.102700.

[13] C. Li, Z. Qu, S. Wang, and L. Liu, "A method of cross-layer fusion multi-object detection and recognition based on improved faster R-CNN model in complex traffic," Pattern Recognit. Lett., vol. 145, pp. 127–134, 2021, doi: 10.1016/j.patrec.2021.02.003.

[14] H. Li et al., "A defense method based on attention mechanism against traffic sign adversarial samples," Inf. Fusion, vol. 76, no. March 2020, pp. 55–65, 2021, doi: 10.1016/j.inffus.2021.05.005.

[15] H. Kwon et al., "NeuroImage Early cortical signals in visual stimulus detection," Neuroimage, vol. 244, no. September, p. 118608, 2021, doi: 10.1016/j.neuroimage.2021.118608.

[16] S. Messaoud, S. Bouaafia, A. Maraoui, and A. Chiheb, "Deep convolutional neural networks-based Hardware – Software on-chip system for computer vision application ☆," Comput. Electr. Eng., vol. 98, no. June 2021, p. 107671, 2022, doi: 10.1016/j.compeleceng.2021.107671.

[17] E. Karaaslan, U. Bagci, and F. N. Catbas, "Attention-guided analysis of infrastructure damage with semi-supervised deep learning," Autom. Constr., vol. 125, no. April 2019, p. 103634, 2021, doi: 10.1016/j.autcon.2021.103634.

[18] K. Nogueira, O. A. B. Penatti, and J. A. dos Santos, "Towards better exploiting convolutional neural networks for remote sensing scene classification," Pattern Recognit., vol. 61, pp. 539–556, 2017, doi: 10.1016/j.patcog.2016.07.001.

[19] Y. Anagun, S. Isik, and E. Seke, "SRLibrary: Comparing different loss functions for super-resolution over various convolutional architectures," J. Vis. Commun. Image Represent., vol. 61, pp. 178–187, 2019, doi: 10.1016/j.jvcir.2019.03.027.

[20] K. Fu, Q. Zhao, I. Yu-Hua Gu, and J. Yang, "Deepside: A general deep framework for salient object detection," Neurocomputing, vol. 356, pp. 69–82, 2019, doi: 10.1016/j.neucom.2019.04.062.

[21] Y. Wang, Z. Fang, and H. Hong, "Comparison of convolutional neural networks for landslide susceptibility mapping in Yanshan County, China," Sci. Total Environ., vol. 666, pp. 975–993, 2019, doi: 10.1016/j.scitotenv.2019.02.263.

[22] B. Zhao, X. Li, X. Lu, and Z. Wang, "A CNN–RNN architecture for multi-label weather recognition," Neurocomputing, vol. 322, pp. 47–57, 2018, doi: 10.1016/j.neucom.2018.09.048.

[23] J. Du, C. Vong, S. Member, and C. L. P. Chen, "Novel Efficient RNN and LSTM-Like Architectures : Recurrent and Gated Broad Learning Systems and Their Applications for Text Classification," pp. 1–12, 2020.

[24] H. Chen, L. Wu, J. Chen, W. Lu, and J. Ding, "A comparative study of automated legal text classification using random forests and deep learning," Inf. Process. Manag., vol. 59, no. 2, p. 102798, 2022, doi: 10.1016/j.ipm.2021.102798.

[25] Z. Parcheta, G. Sanchis-Trilles, F. Casacuberta, and R. Rendahl, "Combining Embeddings of Input Data for Text Classification," Neural Process. Lett., vol. 53, no. 5, pp. 3123–3151, 2021, doi: 10.1007/s11063-020-10312-w.

[26] X. Li, M. Cui, J. Li, R. Bai, Z. Lu, and U. Aickelin, "A hybrid medical text classification framework: Integrating attentive rule construction and neural network," Neurocomputing, vol. 443, pp. 345–355, 2021, doi: 10.1016/j.neucom.2021.02.069.

[27] M. A. Ibrahim, M. U. Ghani Khan, F. Mehmood, M. N. Asim, and W. Mahmood, "GHS-NET a generic hybridized shallow neural network for multi-label biomedical text classification," J. Biomed. Inform., vol. 116, no. April 2020, p. 103699, 2021, doi: 10.1016/j.jbi.2021.103699.

[28] F. Nielsen, "Anticipation-RNN : enforcing unary constraints in sequence generation , with application to interactive music generation," vol. 4, 2018, doi: 10.1007/s00521-018-3868-4.

[29] D. Bisharad, "Music genre recognition using convolutional recurrent neural network architecture," no. April, pp. 1–13, 2019, doi: 10.1111/exsy.12429.

[30] A. Onan, "Sentiment analysis on massive open online course evaluations: A text mining and deep learning approach," Comput. Appl. Eng. Educ., vol. 29, no. 3, pp. 572–589, 2021, doi: 10.1002/cae.22253.

[31] J. L. Huan, A. A. Sekh, C. Quek, and D. K. Prasad, "Emotionally charged text classification with deep learning and sentiment semantic," Neural Comput. Appl., vol. 34, no. 3, pp. 2341–2351, 2022, doi: 10.1007/s00521-021-06542-1.

[32] Á. Arcos-García, J. A. Álvarez-García, and L. M. Soria-Morillo, "Deep neural network for traffic sign recognition systems: An analysis of spatial transformers and stochastic optimisation methods," Neural Networks, vol. 99, pp. 158–165, 2018, doi: 10.1016/j.neunet.2018.01.005.

[33] Á. Arcos-García, M. Soilán, J. A. Álvarez-García, and B. Riveiro, "Exploiting synergies of mobile mapping sensors and deep learning for traffic sign recognition systems," Expert Syst. Appl., vol. 89, pp. 286–295, 2017, doi: 10.1016/j.eswa.2017.07.042.

[34] J. Li, X. Mei, S. Member, D. Prokhorov, and S. Member, "Deep Neural Network for Structural Prediction and Lane Detection in Traffic Scene," pp. 1–14, 2016.

[35] J. Kim, J. Kim, G. Jang, and M. Lee, "Fast learning method for convolutional neural networks using extreme learning machine and its application to lane detection," Neural Networks, vol. 87, pp. 109–121, 2017, doi: 10.1016/j.neunet.2016.12.002.

[36] J. Tang, S. Li, and P. Liu, "A review of lane detection methods based on deep learning," Pattern Recognit., vol. 111, p. 107623, 2021, doi: 10.1016/j.patcog.2020.107623.

[37] A. Shustanov and P. Yakimov, "CNN Design for Real-Time Traffic Sign Recognition," Procedia Eng., vol. 201, pp. 718–725, 2017, doi: 10.1016/j.proeng.2017.09.594.

[38] F. Shao, X. Wang, F. Meng, T. Rui, D. Wang, and J. Tang, "Real-Time Traffic Sign Detection and Recognition Method Based on Simplified Gabor Wavelets," 2018, doi: 10.3390/s18103192.

[39] T. Yang, X. Long, A. Kumar, Z. Zheng, and C. Tong, "Deep detection network for real-life traffic sign in vehicular networks," vol. 136, pp. 95–104, 2018, doi: 10.1016/j.comnet.2018.02.026.

[40] Á. Arcos-García, J. A. Álvarez-García, and L. M. Soria-Morillo, "Evaluation of deep neural networks for traffic sign detection systems," Neurocomputing, vol. 316, pp. 332–344, 2018, doi: 10.1016/j.neucom.2018.08.009.

[41] A. Barodi, A. Bajit, M. Benbrahim, and A. Tamtaoui, "An Enhanced Approach in Detecting Object Applied to Automotive Traffic Roads Signs," in 6th International Conference on Optimization and Applications, ICOA 2020 - Proceedings, Apr. 2020, pp. 1–6, doi: 10.1109/ICOA49421.2020.9094457.

[42] U. A. Nnolim, "An adaptive RGB colour enhancement formulation for logarithmic image processing-based algorithms," Optik (Stuttg)., vol. 154, pp. 192–215, 2018, doi: 10.1016/j.ijleo.2017.09.102.

[43] A. Barodi, A. Bajit, M. Benbrahim, and A. Tamtaoui, "Applying Real-Time Object Shapes Detection to Automotive Traffic Roads Signs," 2020, doi: 10.1109/ISAECT50560.2020.9523673.

[44] N. Ben Romdhane, H. Mliki, R. El Beji, and M. Hammami, "Combined 2d/3d traffic signs recognition and distance estimation," IEEE Intell. Veh. Symp. Proc., vol. 2016-Augus, no. Iv, pp. 355–360, 2016, doi: 10.1109/IVS.2016.7535410.

[45] A. Barodi, A. Bajit, A. Tamtaoui, and M. Benbrahim, "An Enhanced Artificial Intelligence-Based Approach Applied to Vehicular Traffic Signs Detection and Road Safety Enhancement," Adv. Sci. Technol. Eng. Syst. J., vol. 6, no. 1, pp. 672–683, 2021, doi: 10.25046/aj060173.

[46] A. Barodi, A. Bajit, M. Benbrahim, and A. Tamtaoui, "Improving the transfer learning performances in the classification of the automotive traffic roads signs," E3S Web Conf., vol. 234, no. February, 2021, doi: 10.1051/e3sconf/202123400064.

[47] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The German Traffic Sign Recognition Benchmark for the IJCNN'11 Competition," Proc. Int. Jt. Conf. Neural Networks, pp. 1453–1460, 2011, [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6033395.

[48] R. Udendhran, M. Balamurugan, A. Suresh, and R. Varatharajan, "Enhancing image processing architecture using deep learning for embedded vision systems," Microprocess. Microsyst., vol. 76, p. 103094, 2020, doi: 10.1016/j.micpro.2020.103094.

[49] B. S. Rao, "Dynamic Histogram Equalization for contrast enhancement for digital images," Appl. Soft Comput. J., vol. 89, p. 106114, 2020, doi: 10.1016/j.asoc.2020.106114.

[50] A. Bouti, M. A. Mahraz, J. Riffi, and H. Tairi, "A robust system for road sign detection and classification using LeNet architecture based on convolutional neural network," Soft Comput., vol. 24, no. 9, pp. 6721–6733, 2020, doi: 10.1007/s00500-019-04307-6.

[51] T. Sercu and V. Goel, "Dense Prediction on Sequences with Time-Dilated Convolutions for Speech Recognition," no. Nips, 2016, [Online]. Available: http://arxiv.org/abs/1611.09288.

[52] H. C. Shin et al., "Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning," IEEE Trans. Med. Imaging, vol. 35, no. 5, pp. 1285–1298, 2016, doi: 10.1109/TMI.2016.2528162.

[53] M. Gao, Q. Zhang, J. Dong, D. Yang, and D. Zhou, "End-to-end speech emotion recognition based on one-dimensional convolutional neural network," ACM Int. Conf. Proceeding Ser., vol. Part F1481, pp. 78–82, 2019, doi: 10.1145/3319921.3319963.

[54] B. Zoph and Q. V Le, "Searching for activation functions," 6th Int. Conf. Learn. Represent. ICLR 2018 - Work. Track Proc., pp. 1–13, 2018.

[55] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc., pp. 1–15, 2015.