Traffic Sign Recognition Using Neural Networks Useful for Autonomous Vehicles

Jihene REZGUI ‡, Amal HBAIEB ∗†, Lamia CHAARI†, Julien Maryland‡
‡ Laboratoire Recherche Informatique Maisonneuve (LRIMa), Montreal, Canada
∗ Higher Institute of Computer Science and Multimedia of Sfax, University of Sfax, Tunisia
† Digital Research Center of Sfax (CRNS)
Laboratory of Technology and Smart Systems (LT2S), University of Sfax, Tunisia
jrezgui@cmaisonneuve.qc.ca, amelhbaieb@yahoo.com, lamiachaari1@gmail.com, garfieldcod99@gmail.com.

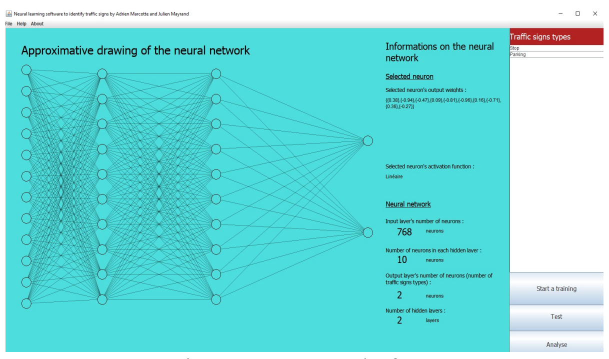
Abstract—**Transportation is a vastly developed research field that tries to optimize humans' safety in everyday life. Recently, autonomous vehicles are being thoroughly developed by a lot of companies such as Uber[1] and Tesla[2], which will improve safety on the roads. These vehicles are using artificial intelligence to correctly interact with their environment. First, we developed a Java program called "ARIBAN" which provides algorithms for recognition and classification of traffic signs with Multi-Layer Perceptron Neural Networks (MLPNN). Second, we use back propagation algorithm in a supervised manner to establish the network. Third, the neural networks are trained and used with a large amount of traffic signs. Finally, the post-processing combines the results to make a recognition decision. Eventually, we have tested our trained network with more than 62 types of traffic signs. Experimental results have demonstrated the effectiveness of the proposed system. Furthermore, the proposed system was deployed within an architecture for autonomous driving.**

Keywords— ARIBAN, Neural Network, Artificial Intelligence, Traffic Signs, Image Recognition, Classification, Autonomous driving.

I. INTRODUCTION

Artificial intelligence (AI) and deep learning have been becoming increasingly popular in the last years as more and more ways to use them are discovered. Traffic Sign Recognition (TSR) is becoming an essential part in Driver Assistance Systems and Automated Driving (ADAS) [3]. Nevertheless, different viewpoints, traffic sign sizes, illuminations and weather conditions, etc. make the TSR task harder. Indeed, existing works stress the TSR decision because of the large variations in visual appearance of traffic sign images. Some studies used optical recognition [4] to classify images using neural networks. Optical recognition is very useful in the context of self-driving cars, as they need to be aware of the presence of nearby obstacles and traffic signs, to be both secure and comply with driving laws. Even if the GPS technology already exists, which can identify the location of traffic signs; it could be useful to use AI in the cases where new traffic signs just were implemented, and to make a double verification of the presence of the traffic signs.

The entire image recognition system is divided into two phases such as training and recognition phase. Both phases include (a) image pre-processing (b) and matrix extraction. Training and recognition phase also include training of the classifier (neuronal network) and its simulation. (a) Pre-processing involves the cropping of the image to get the traffic sign located in the center of the image. (b) After the cropping, the image matrix is normalized into a column matrix containing the values of all pixels of the cropped traffic sign. Then, the normalized image matrix is fed to the neuronal network. It consists of a varying amount of input neurons and as many output neurons as the amount of traffic signs the network was made is shown in Fig. 1.


Neural learning software to identify traffic signs by Adrien Marcotte and Julien Mayrand

Approximative drawing of the neural network

Informations on the neural network

Selected neuron

Selected neuron's output weights :
{{0.38},{-0.94},{-0.47},{0.09},{-0.81},{-0.96},{0.16},{-0.71},{0.36},{-0.27}}

Selected neuron's activation function :
Linéaire

Neural network

Input layer's number of neurons :
768 neurons

Number of neurons in each hidden layer :
10 neurons

Output layer's number of neurons (number of traffic signs types) :
2 neurons

Number of hidden layers :
2 layers

Traffic signs types
Stop
Parking

Fig 1. ARIBAN platform

Our contributions, in this paper, summarized as follows: (1) We introduce our Java platform, called "ARIBAN" [5], which provides many analysis tools to identify and classify road signs; (2) We propose Multi-Layer Perceptron Neural Networks (MLPNN), which can be trained and used in the context of image recognition, notably with traffic signs; (3) We enhance the use of two main algorithms: Feed forward propagation (FFPA) and back propagation (BPA) to detect and classify traffic signs; (4) We evaluate the neural network with different types of image to make a recognition decision; (5) We take into account multiple parameters for the training phase such as the number of iterations, the learning rate, the number of images by group involved in the same BPA; (6) We introduce an autonomous driving use case in which we highlight the exchange information about traffic sign recognition through IoV cooperative system.

The remainder of the paper is organized as follows. Section II reviews the related works in traffic sign detection and classification. Then, section III provides information about the training data used during the learning of our neural network and describes the way the platform operates. In section IV, experiments are conducted to evaluate the ARIBAN algorithms. Finally, conclusions are drawn in Section V.

978-1-7281-4275-3/19/$31.00 ©2019 IEEE

II. Related Work

The new challenges in intelligent transport require more research effort from different perspectives to optimize humans' safety life. Studies in literature are specifically focusing on how autonomous vehicles could avoid obstacles. However, TSR is a major issue to recognize and avoid nearby obstacles.

Tesla Autopilot [6-7] is an advanced driver-assistance system using artificial intelligence to help the driver reach his destination safely. This AI can steer the car, regulate its speed, change lanes and park without any driver input. It uses eight cameras, twelve ultrasonic sensors and a radar to be able to recognize objects under any circumstances so the AI can make the car avoid them. The AI uses machine learning algorithms to be able to improve its efficiency through time. As our ARIBAN platform enforced by ***FFPA and BPA***, it uses many data examples to teach the AI. When drivers use Tesla Autopilot, the system sends data collected to Tesla so the AI always learn more [8]. In the future, cars using Tesla Autopilot will be completely autonomous, being driverless vehicles, using an advanced artificial intelligence. Popular or specially designed schemes include two phases: Detection and Classification to recognize traffic signs in an image. There are many researchers working on this challenging task to propose solutions. However, it is not easy to compare their methods until the release of the German TSR Benchmark [9] and KUL Belgium Traffic Sign Dataset [10]. Proposed methods, to solve detection and classification issues, are reviewed as follows:

A. ***Traffic sign detection*** [11-17]

Because of the harsh environment conditions, the images obtained by the camera are with low quality. Captured image pre-processing can be used to improve the traffic sign regions and to make it easier for subsequent steps. In order to have more distinct signs, the commonly way is transforming images into a new color space. Several color spaces schemes are used, for example Hue Saturation Intensity [11], improved Hue Saturation Lightness [12], normalized color space [13], and histogram of oriented gradients (HOG) [14]. Additionally, using machine learning [15], authors proposed a color probability model to improve the main color of the signs while removing the background regions. Authors trained Support Vector Machine (SVM) classifier [16-17] to map each pixel in the images to a gray value with higher score in the traffic sign. In previous traffic sign detection works, threshold based methods are used [11]. Some other methods like Convolutional Neural Network (CNN) [16] were also used whereas in this paper we used MLPNN.

B. ***Traffic sign classification***[18-25]

Traditional methods for classification include feature extraction and classifier training. Numerous studies enforce SVM classifier with Multi-Layer Perceptron (MLP) with radial histogram features [19] in opposite to our ARIBAN which uses classical MLP. Other combinations with SVM are proposed as HOG features [14], K-d trees and Random Forests with Distance Transforms and HOG features [18], Artificial Neutral Network with Rotation Invariant Binary Pattern based features [20], local image permutation interval descriptor [21]. In [22], three different features representations are extracted: (1) dense scale-invariant feature transform, (2) HOG and (3) Local binary patterns features. Then they were implemented with locality-constraint linear coding. After, the results were generated by spatial pyramid pooling. These features were combined as the final features of a traffic sign, using SVM as the classifier. In [23], Multi-column CNNs, were proposed to classify traffic signs. They train multiple CNNs with different data pre-processing and weight initialization. However, in [24], local and global features in CNN were combined to recognize traffic signs. Compared to [23], an enhanced version of cross entropy loss [25], used in training CNN, obtains better results. Authors in [28] proposed a new TSR approach for intelligent vehicles. The suggested road sign recognition involves three steps; firstly, the input image mapped from the Cartesian coordinate system to the log-polar one. Secondly, from the image represented in the log-polar coordinate system the HOG, the local binary pattern (LBP) and local self-similarity characteristics (LSS) are calculated. Thirdly, their system performs classification using the random forest classifier. They tested their solution on the German Traffic Sign Recognition Benchmark dataset.

We conclude constructing well-designed network architecture and training a practical model are still challenging tasks, even though CNN is proved to be efficient in image classification. In the next section, we present our MLPNN integrated in ARIBAN platform, which recognize and classify traffic signs in images.

C. ***Traffic sign Recognition for Autonomous Driving*** [29-30]

Numerous real-time systems including localization, planning, environment perception, and control should be implemented to accomplish autonomous tasks of a vehicle in urban situations. Autonomous vehicles today depend on the constructed road maps to provide the location of all traffic signs. To automate this process, authors in [29] built a laser-based generic sign detector that locate the orientation and the position of all signs surrounding the autonomous vehicle. Furthermore, they implemented a direction invariant classifier that differentiates stop signs from nonstop signs, even when viewed from the back or side. In their system, they used Haar-type filters with a sliding window to fix the differential along the edges of the target sign. According to the authors, their system detect signs at 89% precision. In [31] authors introduced an Advanced Driver-Assistance Systems (ADAS) as a path toward autonomous vehicles. They confirm that many ADASs support traffic sign detection. The common use case is the identification of the speed limit on the road. An ADAS would alert the driver in case of the vehicle speed is over the limit.

Sign recognition is a primordial task of autonomous vehicles. Any misrecognition of traffic signs can lead to disastrous accident. In [30], authors examined security attacks against TSR systems for Deceiving Autonomous caRs with Toxic Signs (we call the proposed attacks DARTS). They demonstrated a wide range of attacks on TSR. They proposed two methods to create toxic signs. The first one-named Out-of-Distribution attacks allow an adversary to convert any sign or logo into a targeted adversarial example. The second method known as The Lenticular Printing attack allows an adversary to embed a potentially dangerous traffic sign into a safe one, with no access to the internals of the classifier. Our results demonstrate that the proposed attacks are successful under both settings and threat models. They demonstrated the effectiveness of attacks virtual and real-world settings.

III. The ARIBAN Platform

A. ***Platform's Input***

The kind of images which can be used to train or test neural networks on our platform are of the .ppm, .jpg or .png format, of any sizes. They will be scaled to fit to the neural network's specifications.

To train and test the neural networks on our platform, we used the KUL Belgium Traffic Sign Dataset [10]. They consist of 51 different traffic signs, each divided in two categories: training and testing. While the training images are to be used to train the neural network, the testing images are used to test its efficiency. The reason we want the testing images to be different than the training images is that we need to know whether or not the neural network is able to identify any traffic sign, and not just the ones that were used to train it. It is also important to note that the neural networks can be saved and opened for later uses.

B. *ARIBAN's analysis process*

1) ***The neural network creation**:* The first step is to create a neural network with the desired settings. The types of traffic signs that will be analyzed need to be specified, as well as the number of hidden layers, the amount of neurons in those hidden layers and the size of the pictures the neural network can analyze. The size of which the pictures that will be fed to the neural network will be resized needs to be defined, as it will change the amount of neurons in the input layer. The activation function of every layer also needs to be specified, although the function for the last layer is already set to sigmoid, as we want the neural network to produce a number between 0 and 1, 1 being the most certain about a certain type of traffic sign and 0 being the least. The channels of color in which the image fed to the neural network will be decomposed can also be chosen at this point.

2) ***The training phase**:* The second step after creating the neural network is training it. When starting a training session, many parameters need to be specified. First of all, whether to use the ***stochastic*** method and, if the option was chosen, how many training images are used in every set. Second of all, the location of the images used during the training and their types need to be specified, whether by identifying folders containing a certain type or individual images. Finally, the number of epochs needs to be specified: *how many times the network will effectuate the backpropagation technique*?

3) ***The testing phase**:* After training the neural network, it is important to recognize whether or not it has learned correctly. The testing phase has very few parameters to define, which are very similar to the ones in the training phase. Only the location and the types of the pictures that will be used to test the neural network need to be specified. After the testing process is done, the program will output the percentage that the neural network guessed correctly.

4) ***Analysis of a more complex image**:* This phase consists in analyzing a single image which can contain a traffic sign. The image that will be analyzed needs to be selected, and then the program starts the process to find the presence of traffic signs. The program will then tell the user which traffic sign was found in the image.

C. *The neural network's*

Neural networks used in our program are multilayer perceptron composed with an input layer, one or more hidden (intermediate) layers and an output layer. Each layer is composed with neurons, the base unit of the neural network, that is defined by a real number and an activation function, described later. The goal of a neural network is to transform input values into output values. In our case, each input value corresponds to the ***RGB*** value of a pixel in an image, and each output value correspond to a ***different traffic sign***. The goal of that neural network is to guess the presence of traffic signs in the analyzed image. The value of an output neuron corresponds to the degree of certainty of presence of the traffic sign represented by that neuron in the image. Each neuron from a layer is connected to each neuron in the next layer with a real number, called *weight*. Each weight has a random value determined at the initiation of the neural network. Also, each layer can have an extra neuron, called *biais* that will affect the value of the neurons in the next layer. Its goal is to improve the precision of the algorithm by having an unrelated influence in the calculation to the input values.

1) Feed forward propagation algorithm

The first algorithm used in a multilayer perceptron is the ***feed forward propagation*** algorithm, called **FFPA**. This is the main algorithm of the neural network, its goal is to map the input values into the output values. Its first step is to determine the values of the neuron of the first hidden layer. First, each neuron of the input layer sends its value into each neuron of the next layer, multiplied by the value of the weight. Each neuron of the next layer then takes multiple values, one for each neuron of the first layer, and does the sum of each of the values. This new value is sent into the activation function related to the neuron that will transform the value into another value, which is the final value of that neuron. This process is repeated until the last layer to determine the value of each neuron in the network.
There are various types of activation functions, the sigmoid function, the linear function, the hyperbolic tangent function and finally, the hard limiting function. Table 1 illustrates the equation of the different activation function.

TABLE 1: Activation functions

Function Name	**Equation of the activation function**
Sigmoid	$f(x) = 1/(1 + e^{-x})$
Linear function	$f(x) = x$
Hyperbolic tangent	$f(x) = (e^{x} - e^{-x})/(e^{x} + e^{-x})$
Hard limiting	$f(x) = 1 \; if \; x \geq 0 \; f(x) = 0 \; if \; x < 0$

To simplify **FFPA**, it is possible to use matrices operations. Indeed, by placing the input values in a column matrix, and the weights in a normal matrix, the values of the next layer are given by the resulting matrix of the multiplication of these two matrices where each value has passed into the activation function. The values of the neurons of the next layer are described with the next function:

$$H = \sigma(W * X + B) \qquad (1)$$

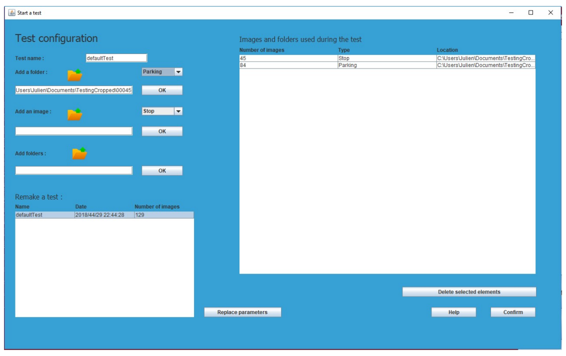
Where H is the Matrix of the values of the next layer, σ is the Activation function of the next layer, W is the Matrix of the weights, X is the Column matrix of the values of the current layer and B is the Matrix of the *biais* neuron of the current layer

With this algorithm, the neural network is able to guess the presence of traffic signs in the image. However, since the weights values are randomly determined at the initiation, the neural network needs to change these weights depending on the traffic signs to analyze to be able to do good recognition. To do this, the network needs to compare the results that it predicted with the **FFPA** with real results. The difference between the objective and the predicted values gives the errors of each output neuron of the network, describing the precision of the algorithm. To have an estimation of this precision, the algorithm does the average of each squared error, which gives the cost function of the network.
The cost function is described as follows:

$$C(t,y) = 1/N \sum_{i=0}^{N} (t[i] - y[i])^2 \qquad (2)$$

Where C(t,y) is the Cost function, N: Number of neurons in the layer, t is theTarget value of the neuron and y is the Predicted value of the neuron.

Since the goal of neural network is to recognize the presence of traffic signs with the fewest mistakes possible, our objective is to minimize this cost function. To do this, it is important to know that the results given by the algorithm are based on the weights between the neurons of the network. Our next step is to find the most appropriate values for these weights to recognize correctly traffic signs in an image. In order to do modify these values, so to minimize the cost function and improve the precision of the neural network, it needs a second algorithm called backpropagation algorithm.


Start a test

Test configuration

Test name : defaultTest
Add a folder : Parking
Users\Julien\Documents\TestingCropped\00045
Add an image : Stop
Add folders :

Remake a test :

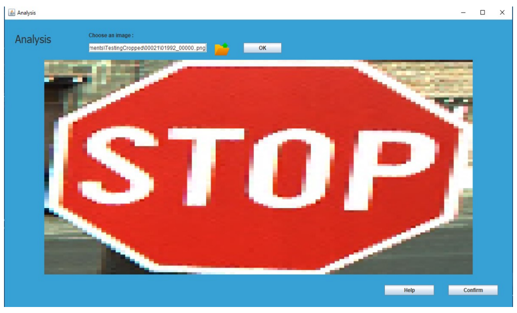
Name	Date	Number of images
defaultTest	2018/44/29 22:44:28	129

Images and folders used during the test

Number of images	Type	Location
45	Stop	C:\Users\Julien\Documents\TestingCro...
84	Parking	C:\Users\Julien\Documents\TestingCro...

Fig.2.Testing phase

In our program, the **FFPA** is used in every step: during the training, the testing and the analysis. It is particularly used in the testing and the analysis parts, where the neural network tries to identify the traffic sign present in the image. In our application, during the testing part as shown in Fig. 2, it is possible to choose the images that will be used to test the efficiency of the neural network.


Analysis

Analysis

Choose an image : ments\TestingCropped\00021\01992_00000..png

Fig. 3. Analysis phase

During the analysis part, it is possible to choose the image to analyze with the present neural network. The application will then tell the user about the identification made by the neural network. (Fig.3).

2) Back propagation algorithm

The second algorithm is named ***back propagation*** **(BPA)**. Its goal is to train the neural network, to improve its precision depending on the traffic signs that need to be recognized by minimizing the cost function. To minimize the error of the network, it needs to find the adapted weights' values to classify correctly the image give in input. To do this, it needs to use the ***gradient descent*** algorithm (**GDA**), a mathematical algorithm that minimize a function using the gradient vector, a vector that describes the directional derivative of a multivariable function oriented in the direction of the maximization of the function at a specific point.

To determine the gradient vector used in this algorithm, it is necessary to find the different variables present in the complete neural network that have an impact on the results.

In our case, the gradient is based on the values of the neurons, on the errors of the network, on the derivative of the activation functions and on the learning rate, which describe the learning speed of the network.

The function describing the variation of the values of the weights in the network is:

$$\Delta W = n * E * \sigma' * X^T \qquad (3)$$

Where ΔW is the Column matrix of the difference between the actual value of the weight and the new value of the weight, n is the Learning rate, E is the Column matrix of the errors of the actual layer, σ' is the Derivative of the activation function,

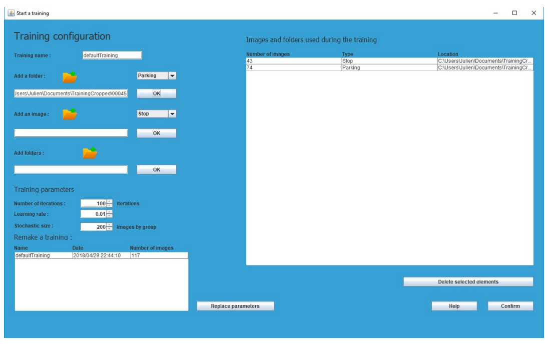
And X^T is the Transposed column matrix of the neuron values of the actual layer.

The **GDA** consists of following the inverse direction of the gradient vector since, by definition, this vector seeks the maximization of the function, and we want the inverse of the vector to minimize the function. After following it for a short distance, it needs to be recalculated to find the new direction of that vector that will minimize the function. With this first step, the values of the weights will be changed, which will reduce the cost function, and improve the precision of the network. To find the relative minimum of the cost function, this step needs to be repeated many times, until the function is minimized. It is possible to compare this algorithm with the descent of a hill to its hollow by taking the shortest way.

It is essential to recalculate the gradient vector after following it for a short distance in is inverse direction, because this vector seeks to minimize the function at the point of the function where it's calculated, and only at that point. The inverse gradient vector does not minimize the function anymore when it is far from its origin point. By travelling short distances in its opposite direction and by then recalculating it, we make sure that it's always heading towards the relative minimum of the function.

To apply ***GDA*** in our neural network, we start with the output values and calculate the values that minimize the error with the weights between the output layer and the last hidden layer using the gradient formula that calculates the change in the weight values. After modifying these first weights, the algorithm does the same step with the weights located between the last hidden layer and the layer that precedes it. That step is repeated until every weight has changed their values, which will have minimized a little bit the cost function. This algorithm is called **BPA**, because it travels the neural network in the opposite direction.

After travelling the neural network one time in its opposite direction, the training method will reproduce these two algorithms with another image, which will give another predicted result with the **FFPA**. The **BPA** will then minimize a little bit more the cost function, which will improve the precision of the neural network. After doing these steps with thousands of example images that contains every type of traffic sign, the **BPA** should have succeeded to reach the minimum of the cost function, which means that it will do minimal errors, and therefore will have learned to recognize traffic signs in an image with a good accuracy.


Start a training

Training configuration

Training name : defaultTraining

Add a folder : Parking

Jsers\Julien\Documents\TrainingCropped\00045 OK

Add an image : Stop

OK

Add folders :

OK

Training parameters

Number of iterations : 100 iterations

Learning rate : 0.01

Stochastic size : 200 Images by group

Remake a training :

Name	Date	Number of images
defaultTraining	2018/04/29 22:44:10	117

Images and folders used during the training

Number of images	Type	Location
43	Stop	C:\Users\Julien\Documents\TrainingCr...
74	Parking	C:\Users\Julien\Documents\TrainingCr...

Delete selected elements

Replace parameters

Help

Confirm

Fig.4. Training Phase

These algorithms are all used while training the neural network as shown in Fig.4., which can be created in our application, where it is possible to choose the images used for the training, the number of iterations, which is the number of times we train the neural network with one group of images, the learning rate, which is the rate that determines the speed of the training and finally, the stochastic size, which determines the number of images used in one group of images.

IV. Platform Results

A. *Simulation Input*

To obtain the results in the next subsection, the following input parameters were specified: the learning rate (we picked two exponentially relative lower values: **LR**=0.01 and **LR**=0.02), the number of iterations (**Ite**=100 and **Ite**=150), the number of images by group (we took of constant value of 50), and the number of hidden layers (from 2 to 5 layers). The main purpose is to determinate the best parameters that increase the recognition accuracy and reduce the processing time. Different training and testing traffic sign images (**TS**) were selected for making performance evaluation (e.g., stop sign, parking sign, speed limit sign, etc.).

B. *Generated Results*

The experiments were repeated twice on the cited database, using a network with two hidden layers that have 12 neurons each. Recognition accuracy averages are summarized in Table 2.

TABLE 2: Recognition *accuracy*

***TS* class number**	***LR*: 0.01 *Ite*: 100**	***LR*: 0.02 *Ite*: 100**	***LR*: 0.01 *Ite*: 150**
2 (Red circle/Blue circle)	99.29%	98.87%	99.42%
3 (Red circle/Blue circle/Triangle)	99.08%	98.30%	99.22%
4 (Stop/Parking/ Triangle/Red circle)	95.65%	94.22%	95.79%
5 (Stop/Parking/ Triangle/Red circle/Blue circle)	93.11%	91.12%	93.20%
6 (Stop/Parking/ Triangle/Red circle/Blue circle/Speed limit)	90.95%	89.03%	91.04%

We notice that we can reach good accuracy results as we train the established network with more number of iterations. Furthermore, a low learning rate can make the training phase more effective (that refers to the shallow weights updates of the network). However, it will increase the training time.

C. *Neural network results*

As aforementioned, we have tested the accuracy rate with a different number of hidden layers (**HL**: varying from 3 to 5 layers, with 12 neurons in each layer) and using the same traffic sign images as above. We used fixed values for learning rate and iterations number parameters (**LR**=0.01, **Ite**=100). The results are shown below (Table 3).

According to the table, we note that increasing the number of hidden layers up from 2 to 5 layers has enhanced the accuracy rate.

D. *Autonomous Driving use case: TSR*

The self-driving vehicle is comprised of a full suite of input components (e.g., sensors), Electronic Control Units (ECUs), actuators, intra-vehicular networks (e.g., Controller Area Network (CAN) [26], Local Interconnect Network (LIN), Ethernet, etc.), and advanced specific sub-systems to perform its functional blocks such as TSR. Sharing information about traffic signs among autonomous vehicles like no-overtaking sign or speed limit sign is significant for safe guidance and navigation. Accordingly, this subsection presents a traffic sign information exchange for autonomous driving within an IoV cooperative system. The data flow can be accomplished with an In-Car Gateway communication as proposed in our previous work [27]. The overall system architecture is shown in Fig. 5.

TABLE 3: Recognition *accuracy*

***TS* Class number**	***HL number*: 3 *LR*: 0.01 *Ite*: 100**	***HL number*: 4 *LR*: 0.01 *Ite*: 100**	***HL number*: 5 *LR*: 0.01 *Ite*: 100**
2 TSC	99.42%	99.50%	99.67%
3 TSC	99.15%	99.24%	99.36%
4 TSC	95.71%	95.79%	95.88%
5 TSC	93.18%	93.25%	93.32%
6 TSC	91%	91.05%	91.10%
7 TSC	87.73%	87.78%	89.80%

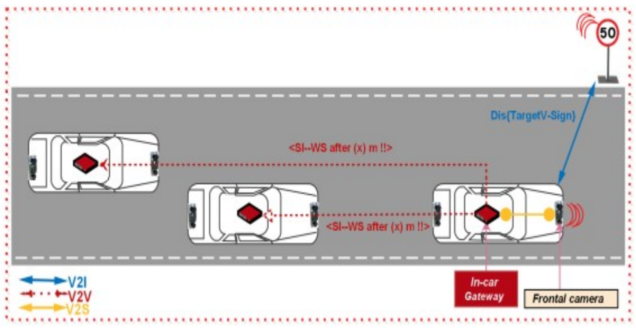


Fig.5. TSR environment architecture

As depicted, the driverless car zone involves a mounted In-Car Gateway device which will serve as the coordinator between the different collaboration system components and provides required communications (i.e., Vehicle-to-Infrastructure (V2I), Vehicle-to-Vehicle (V2V), and Vehicle-to-Sensor (V2S) communications in our case).

The overall scenario description goes as follows: the target vehicle relies on its front camera to capture upcoming traffic signs. This camera is attached to the TSR subsystem, which will apply our

MLPNN algorithm to interpret perceived signs. Once a traffic sign is recognized (e.g., speed limit icon), vehicle internal systems will act with the proper way to follow the current navigation restrictions such as the Adaptive Cruise Control (ACC) sub-system that will change the appropriate vehicle velocity and maintain required safety distance. We can also calculate the traffic sign distance from the sensing camera to carry out more accurate vehicle components management.

Through the embedded gateway, the target vehicle will likewise, share the sign notification information with its neighboring vehicles, so that all the traveling vehicles will have a continuous reference for the zone driving rules.

It is worth noting that we need a common consensus for the traffic signs information, to facilitate their exchange.

V. Conclusions and future work

In this paper, we have proposed a new Java platform called "ARIBAN" for traffic sign recognition and classification using multi-layer perceptron neural networks. Feed forward propagation back propagation algorithms have been applied to build the network. Then, the neural network has been trained with several various traffic signs. Recognition parameters like learning rate and layers numbers have been taken into account to evaluate the proposed program. Accuracy results have shown the effectiveness of our platform on the KUL Belgium Traffic Sign Dataset.

We will integrate this system with our proposed system for pedestrian detection [32] toward autonomous driving. However, much more work is required to attain fully autonomous driving.

Acknowledgment

This research was financially supported by the “Fonds Québécois de la recherche sur la nature et les technologies (FRQNT).” We would like to thank Adrien Marcotte, student at College Maisonneuve, for his valuable comments and participation in this project.

References

[1] https://www.techradar.com/news/uber-self-driving-cars, [Accessed: 22- July - 2019].

[2] https://www.tesla.com/en_CA/autopilot?redirect=no, [Accessed: 22- July- 2019].

[3] R. Timofte, V. A. Prisacariu, L. J. Van Gool, and I. Reid, "Chapter 3.5: Combining traffic sign detection with 3d tracking towards better driver assistance" in Emerging Topics in Computer Vision and its Applications, C. H. Chen, Ed. World Scientific Publishing, September 2011.

[4] http://www.legalscans.com/ocr.html [Accessed: 22- July- 2019].

[5] J. Rezgui, J. Mayrand, A. Hbaieb and A. Marcotte, "ARIBAN platform", https://github.com/ResearchProjectTSR/ARIBAN-platform, Accessed: 22- May- 2019].

[6] https://www.predictiveanalyticstoday.com/top-companies-autonomous-cars-self-driving-car/[Accessed: 22- July - 2019].

[7] http://www.businessinsider.com/tesla-autopilot-functions-and technology-2017-12. [Accessed: 22- July - 2019].

[8] https://www.techrepublic.com/article/teslas-autopilot-the-smart-persons-guide/[Accessed: 22- July - 2019].

[9] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The German traffic sign recognition benchmark: A multi-class classification competition, " in Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN), Jul. 2011,pp. 1453–1460.

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

[11] A. de la Escalera, J. M. Armingol, and M. Mata, “Traffic sign recognition and analysis for intelligent vehicles,” *Image Vis. Comput.*, vol. 21, no. 3, pp. 247–258, 2003.

[12] H. Fleyeh, “Color detection and segmentation for road and traffic signs,” in *Proc. IEEE Conf. Cybern. Intell. Syst.*, vol. 2. Dec. 2004, pp. 809–814.

[13] W. Ritter, F. Stein, and R. Janssen, “Traffic sign recognition using colour information,” *Math. Comput. Model.*, vol. 22, nos. 4–7, pp. 149–161, 1995.

[14] J. Greenhalgh and M. Mirmehdi, “Real-time detection and recognition of road traffic signs,” *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 4, p. 1498–1506, Dec. 2012.

[15] Y. Yang, H. Luo, H. Xu, and F. Wu, “Towards real-time traffic sign detection and classification,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 7, pp. 2022–2031, Jul. 2016.

[16] Y. Wu, Y. Liu, J. Li, H. Liu, and X. Hu, “Traffic sign detection based on convolutional neural networks,” in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Aug. 2013, pp. 1–7.

[17] M. Liang, M. Yuan, X. Hu, J. Li, and H. Liu, “Traffic sign detection by ROI extraction and histogram features-based recognition,” in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Aug. 2013, pp. 1–8.

[18] F. Zaklouta, B. Stanciulescu, and O. Hamdoun, “Traffic sign classification using K-d trees and Random Forests,” in *Proc. Int. Joint Conf.Neural Netw. (IJCNN)*, Jul. 2011, pp. 2151–2155.

[19] Y. Jiang, S. Zhou, Y. Jiang, J. Gong, G. Xiong, and H. Chen, “Traffic sign recognition using ridge regression and OTSU method,” in *Proc. IEEE Intell. Veh. Symp. (IV)*, Jun. 2011, pp. 613–618.

[20] S. Yin, P. Ouyang, L. Liu, Y. Guo, and S. Wei, “Fast traffic sign recognition with a rotation invariant binary pattern based feature,” *Sensors*, vol. 15, no. 1, pp. 2161–2180, 2015.

[21] T. Tian, I. Sethi, and N. Patel, “Traffic sign recognition using a novel permutation-based local image feature,” in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2014, pp. 947–954.

[22] Y. Zhu, X. Wang, C. Yao, and X. Bai, “Traffic sign classification using two-layer image representation,” in *Proc. 20th IEEE Int. Conf. ImageProcess. (ICIP)*, Sep. 2013, pp. 3755–3759.

[23] D. Ciresan, U. Meier, and J. Schmidhuber, “Multi-column deep neural networks for image classification,” in *Proc. IEEE Conf. Comput. Vis.Pattern Recognit. (CVPR)*, Jun. 2012, pp. 3642–3649.

[24] P. Sermanet and Y. LeCun, “Traffic sign recognition with multi-scale convolutional networks,” in *Proc. Int. Joint Conf. Neural Netw. IJCNN)*, Jul. 2011, pp. 2809–2813.

[25] J. Jin, K. Fu, and C. Zhang, “Traffic sign recognition with hinge loss trained convolutional neural networks,” *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 1991–2000, Oct. 2014.

[26] Chaari, L., Masmoudi, N., & Kamoun, L. (2002, October). Electronic control in electric vehicle based on CAN network. In *Systems, Man and Cybernetics, 2002 IEEE International Conference on* (Vol. 7, pp. 4-pp). IEEE.

[27] Hbaieb, A., Rhaiem, O. B., and Chaari, L , “In-car Gateway Architecture for Intra and Inter-vehicular Networks”. In 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC) (pp. 1489-1494). IEEE. (2018).

[28] Ayoub Ellahyani, Mohamed Ansari, Redouan Lahmyed, Alain Trémeau. “A new traffic sign recognition method for intelligent vehicles”, Journal of the Optical Society of America , In press.

[29] Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D., Kammel, S., ... & Sokolsky, M. (2011, June). Towards fully autonomous driving: Systems and algorithms. In 2011 IEEE Intelligent Vehicles Symposium (IV) (pp. 163-168). IEEE.

[30] Sitawarin, C., Bhagoji, A. N., Mosenia, A., Chiang, M., & Mittal, P. (2018). Darts: Deceiving autonomous cars with toxic signs. arXiv preprint arXiv:1802.06430.

[31] Kukkala, V. K., Tunnell, J., Pasricha, S., & Bradley, T. (2018). Advanced Driver-Assistance Systems: A Path Toward Autonomous Vehicles. IEEE Consumer Electronics Magazine, 7(5), 18-25.

[32] Amal HBAIEB, Jihen REZGUI, Lamia CHAARI FOURATI, Pedestrian Detection for Autonomous Driving within Cooperative Communication System, IEEE Wireless Communications and Networking Conference, 15-18 April 2019 // Marrakech, Morocco.