The International Journal of Engineering and Science (IJES)
|| Volume || 9 || Issue || 07 || Series I || Pages || PP 07-19 || 2020 ||
ISSN (e): 2319-1813 ISSN (p): 20-24-1805

Traffic signs detection and identification using NN and CNN

E. Kolberg
Faculty of Engineering, Bar-Ilan University, Israel

Date of Submission: 18-06-2020 Date of Acceptance: 06-07-2020

I. INTRODUCTION

In recent years we face a substantial research efforts towards realization of autonomous cars [1]. The regulation in various countries has strict demands that the autonomous car will be safe at least as a car driven by a common driver is [2]. One stratum is developing an algorithm that gathers data from the environment and cause the car to act accordingly. Among various data types, the detection and identification of traffic signs is of major importance [3]. Since traffic signs enables proper driving, avoiding accidents, alert the driver of dangers along the route, etc., it is critical regarding car control.

Traffic signs identification is a complex challenge [4]. The regulation requirement is having identification close to 100% identification since any false or no identification has a potential for a sudden break at the best or fatal scenario in the worst case [5]. The technology challenge the algorithm has to overcome integration of many variables that influence on the traffic signs condition like faded color, different angles of view, in various lighting conditions, and different weather conditions [4]. Even in the toughest conditions the identification percentages must remain high. On top of that it is required to identify any traffic sign in real-time within a very short time.

There are various techniques that can be used in order to identify traffic signs in a given image. One option is to apply classic image processing techniques [6]. Here there is a need to classify each sign through its shape, colors, its probability to appear in different environments, etc. Once the data is available for the algorithm, techniques like Houph transform, Canny edge detection, and more can be used. These techniques can help in locating the traffic sign localization within the image.However, there is a drawback in using the above mentioned techniques [6]. Various weather conditions, faded signs and similar color objects (e.g. blue sky and blue traffic sign) reduce substantially the identification ability [3]. In addition the time it takes to identify a traffic sign even in ideal conditions is not fast enough. Studies have shown that such a decrease arose because of too strong variations in the illumination, contrast, and angle of rotation in images of localized traffic signs [7], [8].Image processing and SVM methods are described in [9] and [10]. They have a limit regarding precision. A summary of current methods about traffic signs identification is presented in [11]. Each method has advantages and drawbacks. The classic image processing methods has limitations.

Alternative method is machine learning technique. In this paper we present a technique that integrates classic image processing with deep learning. It proved to be efficient and fast.

Methodology and tools

We used machine learning methods for the task of traffic signs' identification. The task is composed of two stages.

First stage: **Detection** - Extracting the traffic sign(s) from the image. It was done by an algorithm that perform categorization using the cascade method and implement a variation of Viola-Jones algorithm. The machine went through a training that included a large set of positive examples (relevant - included at least one traffic sign) and negative examples (non-relevant – include no traffic sign).

Second stage: **identification** – the ability to identify a specific traffic sign out of possible sign list in the image part that was extracted from the whole image.

We tested several techniques for an optimal solution. Deep learning technique presented best results.

<u>Deep learning technique</u>

Deep learning (DL) is based on neuron networks (NN) with various numbers of layers, which allows for different representations of the data.

Neural networks are composed of large number of simple processing units that are called "neurons" (each neuron is similar in its essence to a single Logistic Regression (LR)). The neurons are placed in layers and connected hierarchically.The network is composed of three layer types. Each of this connections is assigned with a certain weight. The weight determines how relevant is the data that goes through the connection, and whether the network should use this data in order to solve the problem.

- The first layer is the input layer that receives data to the network. Each cell in this layer includes one input. The input vector is the input to the network. The number of cells is identical to the input vector length. Each junction in the input (first) layer represents different feature.
- The middle layer is known as the hidden layer (the network might include more than one hidden layer). Each cell in this layer(s) has multiple inputs, usually as many as the number of input cells (Fully Connected). However, there are type of networks that for them this is not true (like CNN networks that we will discuss later).
- The last layer is the output layer which return the processed data as an output. The junctions in each layer are fully connected to the junctions in adjacent layers through direct connection among the neurons.Each cell in this layer has several inputs, usually as many as the number of hidden layer cells (Fully Connected). The vector of the cells' outputs in this layer is the output vector of the network. The number of cells is identical to the number of the output possibilities. The output (last) layer represents the solution tothe problem.

The number of weights in neuron networks is bigger than their number in LR, due to multiple layers situation. Thus, in the training process it will be necessary to train larger number of weights, which will require more time and processing resources on one hand, but it will improve the network or machine abilities in the other hand.Figure 1 presents an example of a typical neuron network.

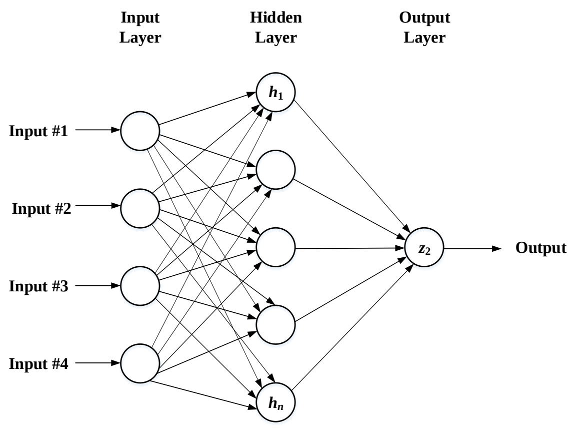


Fig. 1: An example of a typical neuron network.

The network is composed of large number of neurons, as can be seen. Therefore each h_i is like an output of a logistic regression so the mathematical expression will be identical to the one of LR:

$$h_i = g\left(w_{1i}x + b_{1i}\right) = \frac{1}{1+\exp\left(-w_{1i}x - b_{1i}\right)} \tag{1}$$

After receiving all of the h_n's, the algorithm will perform additional LR in order to get the answer in the network output:

$$\hat{y} = p\left(y=1 \middle| x; w, b\right) = \frac{1}{1+\exp\left(-z_2\right)} \tag{2}$$

The classier received in equation (2) is not necessarily linear and it can successfully handle different data types. For example, in order to create a classifier for AND and OR functions, it is possible to use a simple LR. However, in order to create a classifier for XOR function we will have to use neuron network. Figure 2 presents the above mentioned.

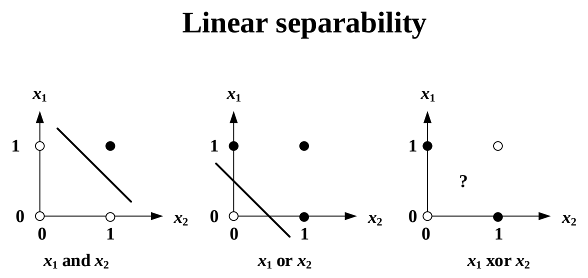


Fig. 2: AND, OR and XOR linear separability

Figure 2 shows that XOR function, as opposed to AND and OR functions, is non separable with a linear separator. This is the reason why LR will not work here. In comparison to the above mentioned method, with a simple neuron network the data can be represented in a different way and thus can be separated linearly as presented in Figure 3.

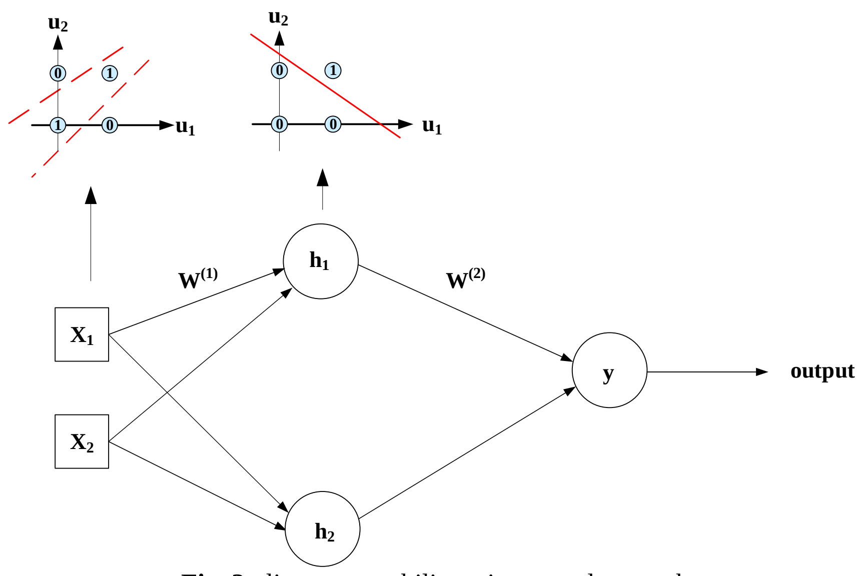


Fig. 3: linear separability using neural network

Next step will be finding the weights. Similar to LR, examples will be used in order to compute the optimal weights. The function that should be maximized similar to LR, is:

$$S = \sum_t \log p\left(y_t \middle| x_t; w, b\right) \quad (3)$$

This function is not convex, which means that it might have several local maxima. That is why there is no guarantee that with Gradient Ascent (GA) method there will be a conversion towards the global maximum point, as is desired. In order to have a better chance to converge to the global maximum point, the GA algorithm will run several times with different weights initialization each time. The maximum of all these runs will be considered as the global maximum.

Neural networks use Feed Forward- Back Propagation (FF-BP) algorithm in order to calculate the derivatives effectively. The algorithm first propagate the data from the beginning of the network up to its end (FF) and then back propagation update (BP). The algorithm use the chain rule which allows for calculating all of the desired derivatives in one pass. Mathematically it will be implemented as follows.

Feed Forward:

$$\begin{aligned}
&\bullet\ z_{1i} = w_{1i}x + b_{1i}\\
&\bullet\ h_i = g\left(z_{1i}\right)\\
&\bullet\ z_2 = w_2 h + b_2\\
&\bullet\ \hat{y} = p\left(y=1 \middle| x; w,b\right) = \frac{1}{1+e^{-z_2}}\\
&\bullet\ S\left(w,b\right) = \log p\left(y=1 \middle| x; w,b\right)
\end{aligned} \tag{4}$$

Back Propagation:

$$\begin{aligned}
\frac{\partial S}{\partial w_2} &= \frac{\partial S}{\partial z_2}\frac{\partial z_2}{\partial w_2} = \left(y-\hat{y}\right)h\\
\frac{\partial S}{\partial w_{1i}} &= \frac{\partial S}{\partial z_2}\frac{\partial z_2}{\partial h_i}\frac{\partial h_i}{\partial z_{1i}}\frac{\partial z_{1i}}{\partial w_{1i}} = \left(y-\hat{y}\right)w_{2i}g'\left(z_{1i}\right)x = \left(y-\hat{y}\right)w_{2i}h_i\left(1-h_i\right)x
\end{aligned} \tag{5}$$

Here LR is performed twice and hence the equations are identical to LR equations. During the BP stage it will be easy to calculate the derivatives. After that GA equations will be used in order to update the new weights:

$$W_{t+1} = W_t + \varepsilon \frac{\partial S}{\partial W}\left(W_t\right) \tag{6}$$

We will add that also in this case it is possible to generalize for taking care of several output cases (Multiclass).

<u>Convolutional Neural Network (CNN)</u>

Feed-forward:

- $z_{1i} = w_{1i}x + b_{1i}$
- $h_i = g(z_{1i})$
- $z_2 = w_2 h + b_2$
- $\hat{y} = p(y=1|x;w,b) = \frac{1}{1+\exp(-z_2)}$
- $S(w,b) = \log p(y|x;w,b)$

Back-propagation:

$$\begin{aligned}
\frac{\partial S}{\partial w_2} &= \frac{\partial S}{\partial z_2}\frac{\partial z_2}{\partial w_2} = (y-\hat{y})h\\
\frac{\partial S}{\partial b_2} &= \frac{\partial S}{\partial z_2}\frac{\partial z_2}{\partial b_2} = (y-\hat{y})\\
\frac{\partial S}{\partial w_{1i}} &= \frac{\partial S}{\partial z_2}\frac{\partial z_2}{\partial h_i}\frac{\partial h_i}{\partial z_{1i}}\frac{\partial z_{1i}}{\partial w_{1i}} = (y-\hat{y})w_{2i}g'(z_{1i})x = (y-\hat{y})w_{2i}h_i(1-h_i)x
\end{aligned} \tag{7}$$

With different architecture, it will be possible to reduce significantly the number of weights. For a certain layer that has an input vector of the size of *m* and output vector of the size *n*, and for a filter with a size of *k*, there will be $k \times n$ weights and computation time of $o\left(k \times n\right)$ with *k* significantly smaller than *m*.

A convolutional neural network promote two central principles that help with machine learning: weights sparsity, and weights sharing.

The integration of all stages to one deep network is presented in figure 4. It includes two convolution layers and then flattening to one vector that is fed to the last layer. The output present the classification.

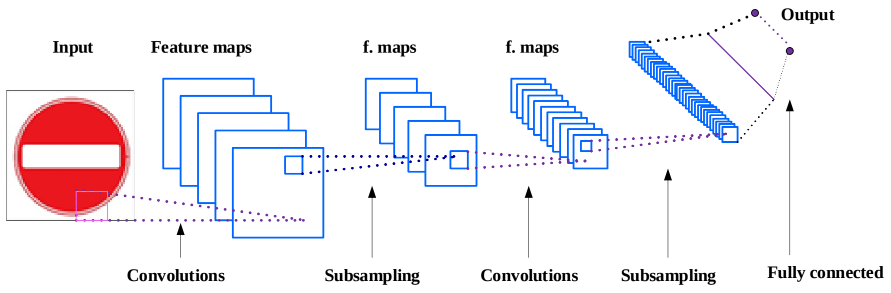


Fig. 4: two convolution layers and flattering process

<u>Data Collection</u>
Machine learning methods require many data samples. In this research we focused on five major traffic signs, which are presented in figure 5.

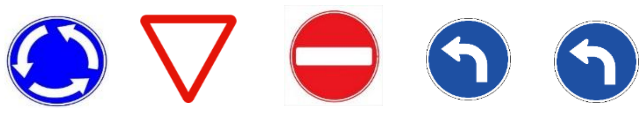


Fig. 5: Major traffic signs

We used a data base with 5500 images of these traffic signs. All images were transferred into 32 x 32 pixels. This size allowed for optimal processing. The database includes many faded signs, different angles of rotation, different lighting conditions, etc. Examples of some images from the database are presented in figure 6.

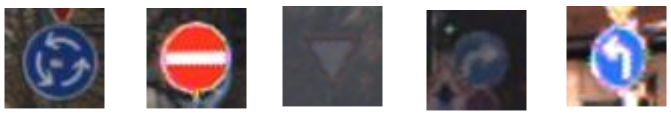


Fig. 6: Images from the database

All images were converted into Grayscale format, in which each pixel is represented by a single number between 0 and 255.The reasons for that are less sensitivity to lighting conditions, the need to have good identification in night conditions, identifying fade traffic signs, less data to deal with compared to color images which will decrease processing time. We assumed that the machine will be good enough to identify traffic signs according to their shape (frame and internal shapes). Examples of images after conversion to gray scale format are presented in figure 7.

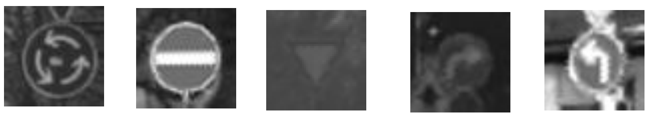


Fig. 7: Images after conversion into Grayscale format

Since there were different number of images for the various traffic signs, we performed simple augmentation actions on the images in the database. This allowed for increasing the database with no need of additional images. It increased the database to have 8000 images. All actions performed were logical and realistic. An example to a specific action of careful rotation is presented in figure 8.

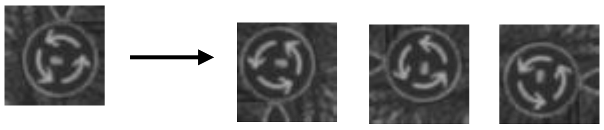


Fig. 8: Different image of the same traffic sign obtained by rotation

The images histogram is presented in figure 9. The histogram shows non uniform distribution. It might introduce slightly different performances for different traffic signs, but it will not be substantial.

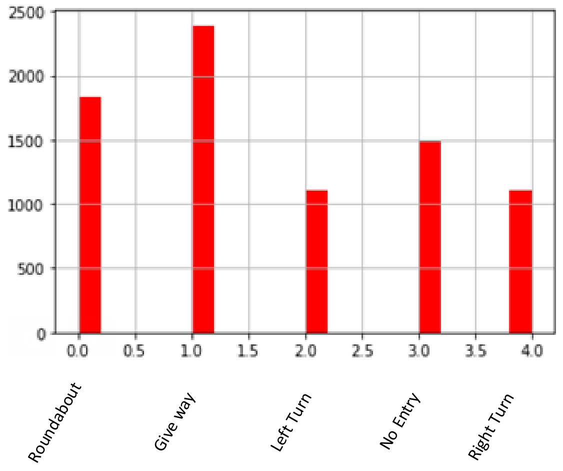


Fig. 9: Histogram of traffic sign images

Traffic sign detection

The input to the algorithm is a video stream from the camera mounted on a car. The images are analyzed sequentially. In order to decrease the detection time and reduce erroneous detections, the algorithm trims 30 percent of the frames' lower side.

The signs detection use a variation of Viola-Jones algorithm [12].

The process is composed of several stages. The filter windows first examine the images by passing all over the image. Each filter is focused on a different feature that the algorithm looks for in the object. When a specific filter determine "negative" in a certain window, this area is now out of the search zone. The other filters will not examine this area anymore. If the filter identifies the feature it is responsible for, this area goes to next filter examination, and so forth.This way the background will be identified in early stages of the examination and will not be tested by Adaboost algorithm in advanced stages.

In the training phase it is desired to have many "positive" images, meaning images that contain the object that is needed to be found, and many "negative" images, meaning images with no relevant objects. Our database includes 8000 "positive" images and about 10000 "negative" road images. The idea is presented in figure 10.

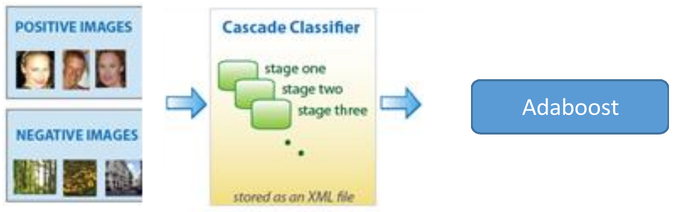
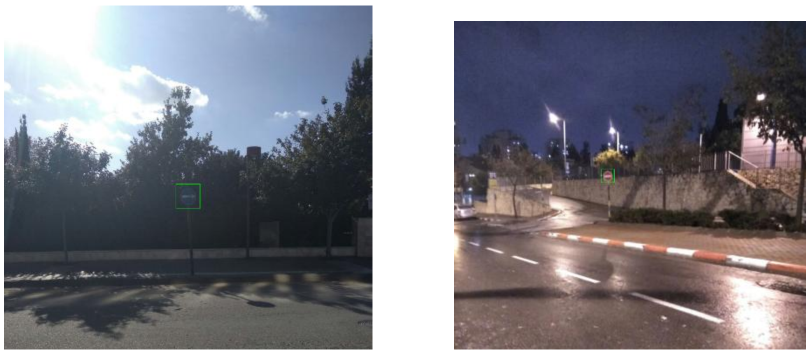


Fig. 10: The detection process

Examples of traffic sign detection are presented in figure 11.



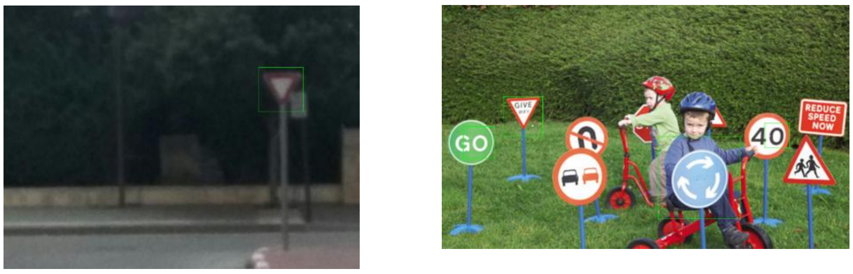


Fig. 11: Traffic signs detection

The images of figure 11 show high percentage of sign detection. It keeps high detection rate even with tuff lighting conditions (night, against sun light), bad image quality, and with different distances from the signs. Even if some false positive signs detection noise happens, the traffic signs identification algorithm will filter them.

<u>Traffic sign identification</u>

In the detection phase, the algorithm isolated the traffic sign from the image. Now the algorithm has to identify the sign. If it identifies the sign as a traffic sign, it should decide which traffic sign it is. If it is an object that is similar but not a real traffic sign, no further process is needed.

The machine learning methods we tested included those of categorization learning. We looked for the method with the highest identification rate within high speed processing.

First we used a relatively naïve method which is a variation of Supervised K-Means algorithm. Given many samples that are presented in a space, the algorithm finds an average point for every category. Then when a new point is received, the algorithm check which is the closest average, and according to this average the algorithm will determine its category. In order to implement this method, for every category each traffic sign was presented by a vector with the size of 32 x 32 = 1024. The average of all these vectors is the average vector for the specific category. When a new sign needed to be identified, the algorithm tested which is the closest average point (vector). The closest point determined the class of the new sign. It yield 51% success rate.

Next we tested *k-nn* algorithm for different *k* values. Given representation in space of all samples, the algorithm needs to take a decision about the new point category. The algorithm examine the k closest neighbors to the new point and assign the point with the category that most neighbors are related to. For example, if k=5 and within 5 closest neighbors, one is of category 2, another one is of category 3, and other 3 neighbors are of category 1, the algorithm will assign the category 1 for the new point. For different k values it turned out that the optimal k is k=1. It yielded 80% correct identifications. It is not good enough.

Next we tested the Generalized Method of Moments (GMM). It uses clustering analysis which assumes that all samples can be related to different Gaussian distributions in space. The Gaussian distributions represent the different categories, and each Gaussian has a specific probability to appear. Since in our case the algorithm knows how to relate each sample group to a certain category, we used Supervised GMM method.

The algorithm computed the expectancy and variance for all samples of each traffic sign. The resulted data created Gaussian distribution that represent the specific traffic sign. In addition the algorithm computed the frequency of specific traffic sign out of all samples which resulted in the probability of each Gaussian to come out.

In the test phase, in order to determine to which category new traffic sign (vector) belongs to, the algorithm checked to which Gaussian the sign fits mostly. Then the algorithm calculated the likelihood to have this sign vector in that specific Gaussian.

The vector was attributed to the Gaussian with the greatest probability product. Each Gaussian represents a specific category.

Using this method yielded 86% successful identifications. It is not enough for traffic signs identification. Next we tested the Logistic Regression method. In this method, each traffic sign is a separate category in itself. The purpose is to find the optimal weights. The input receives a flatted image represented by a vector of length 784 and the output produce a vector of length 5. Each cell of the vector has the probability that the traffic sign in the input is the traffic sign presented by the specific cell. The correct traffic sign should produce high percentage identification. The number of weights is: 784 x 5 = 3920. To this number the algorithm adds 5 bias weights, so in total there are 3925 weights in the model. The identification rate with this method was approximately 92%. It is still not enough concerning safety in autonomous vehicles.

Next we moved to deep learning methods. As we mentioned before, it is based on neuron like networks models which use Logistic Regression models that are connected to each other. We implemented the network with Python Keras library. The network was trained with all the traffic signs data.

The network included two hidden layers. Its input is a vector with size of 784, which is a flatted image (matrix). Its output is a vector with size of 5, where in each cell there is the probability that the input traffic sign is the one that each cell represents. The network diagram is presented in figure 12.

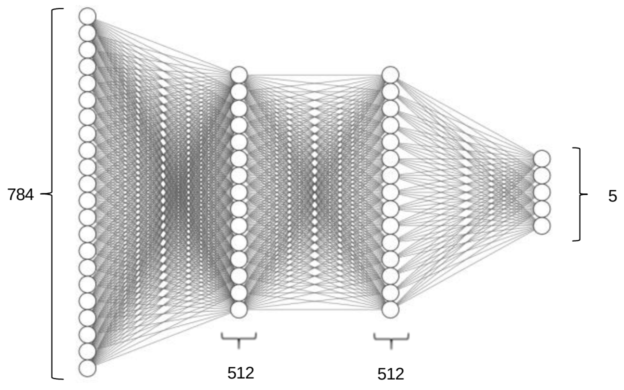


Fig. 12: The Neural Network diagram

The initialization of weights is important in order to receive high performances of the network. We used a common initialization where the weights randomly picked from $[-w, w]$, with $w = \left(\frac{6}{fanIn + fanOut}\right)^2$. *fanIn* is the number of element in the input for each layer. In our example, for the first layer this number will be 784. *fanOut* is the number of elements in the output from each layer. In our example for the second layer it will be 512. The total parameter number is about 670,000 as appear in table 1.

Table 1: Parameters of the different layers

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 512)               401920
_________________________________________________________________
dense_5 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_6 (Dense)              (None, 5)                 2565
=================================================================
Total params: 667,141
```

Logistic Regression method used about 4,000 weights. With Neural network method use more than 165 times that number of weights. The learning rate of the network is $\varepsilon = 0.001$.
The machine accuracy as a function of epochs number is presented in figure 13.

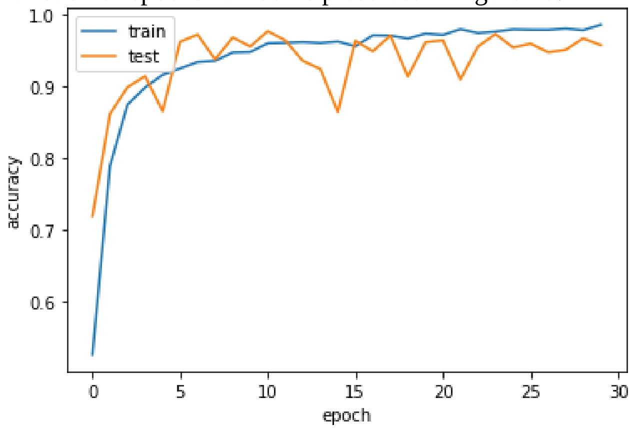


Fig. 13: Accuracy as a function of epochs number

From figure 13 it is clear that that at the beginning of the learning there are low performances but very fast the performances accuracy increase. As number of epochs during training, increase, so is the test accuracy.It is reasonable since with each additional epoch the algorithm converge towards the score's minimum, according to Gradient Ascent method. This is reflected by netter accuracy over the training data. It means that the machine learn the data better with each work through the entire training dataset.
At the end of the training, the test converges to 97% accuracy. It is a good figure but we tried to improve it by using Convolutional Neural Network (CNN). The CNN structure is presented in figure 14.

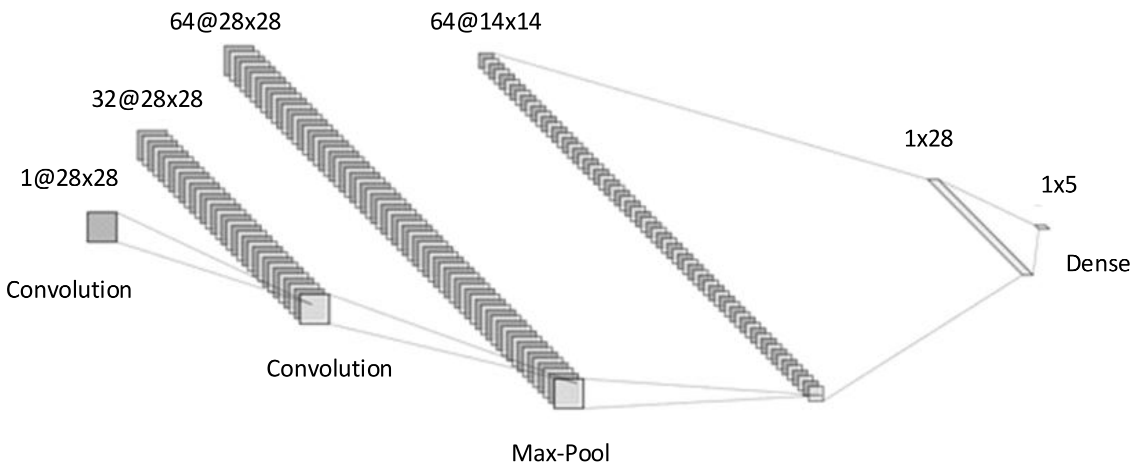


Fig. 14: CNN structure

The weights initialization is identical to the one in regular neural network. The learning constant in this case is $\varepsilon = 0.001$ and is also identical to the one in regular neural network. The parameters number in the network is about 1,600,000 as presented in table 2.

Table 2: Parameters of the CNN layers

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 28, 28, 32)        320
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 28, 28, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 12544)             0
_________________________________________________________________
dense_21 (Dense)             (None, 128)               1605760
_________________________________________________________________
dense_22 (Dense)             (None, 5)                 645
=================================================================
Total params: 1,625,221
```

We trained the network and run tests. The machine demonstrated high performance rate and identified 99% of correct identification rate and only 1% error. While it is true that the CNN has greater number of weights but it is the trade-off in order receive better performances.
The run time of both network types (NN and CNN), after training of traffic sign identification, was fast. For NN it was of order of hundredth of a second and for CNN it was about one tenth of a second.
The machine accuracy as a function of epochs number is presented in figure 15.

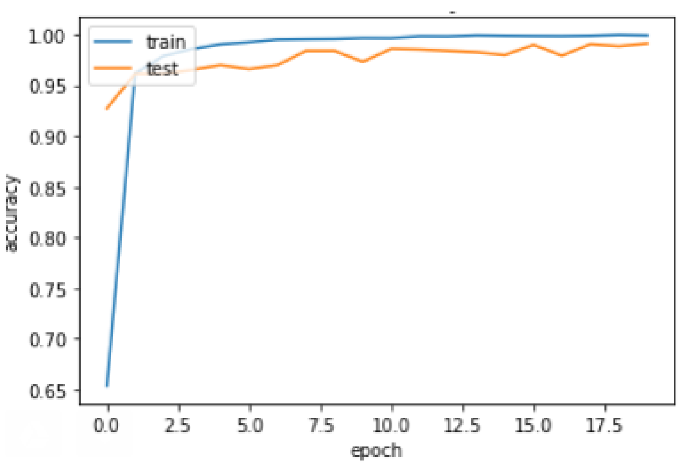


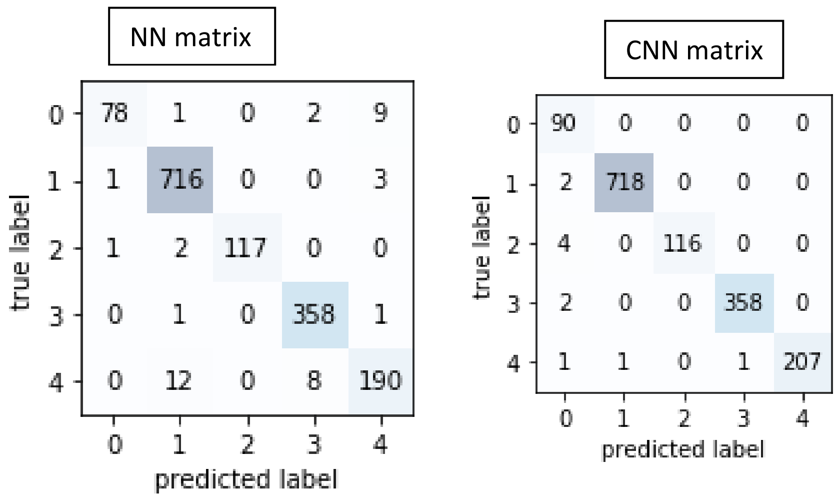
Fig. 15: Accuracy as a function of epochs number in CNN

Observing figure 15 we can see that it has better and faster learning that is expressed by sharper line and by the fact that the conversion takes less epochs (each epoch takes more time in this machine though). The reason for this stems from the relatively high amount of parameters, and the network structure.
Comparison between NN and CNN in our case
In order to compare between these two networks we will define two parameters: precision and sensitivity. It is define for each class separately.

$$precision = \frac{\#\,predictedTrue}{\#\,actualTrue} \tag{8}$$

$$sensitivity(recall) = \frac{\#\,predictedTrue}{\#\,totalPredicted} \tag{9}$$

Sensitivity is defined as the number of times where the machine correctly identified a specific class out of the total number of identification of this class (correct and erroneous).
Up to now we tested the precision of a classifier for all the data set. Now we will define precision for a specific class. Precision in this case is the number of times where the machine correctly identified a specific class out of the total number where this class was present.
In order to find the precision and sensitivity of each classifier, the data was organized in a Confusion matrix, which allows for easy computation of these parameters. It is presented in figure 16.



#	Tr. sign
0	Roundabout (circle)
1	Yield (give way)
2	Left turn
3	Do not enter (noEntry)
4	Right turn

Fig. 16: NN and CNN true vs. predicted labels

Let's take for example traffic sign number 0 (Roundabout). From the matrices we find that CNN has a precision of $\frac{90}{90}=1$, which is 100%. It means that each time a Roundabout sign appeared in the CNN input, it classified it correctly. On the other hand, the NN 12 (9+2+1) different times classifies the Roundabout sign as a different traffic sign when it appeared in its input. The NN precision in this case is $\frac{78}{78+1+2+9}=0.86$, meaning 86%.

Now let's examine the sensitivity parameter. Here CNN has lower sensitivity than the NN. The CNN sensitivity is: $\frac{90}{90+2+4+2+1}=0.9$ meaning 90% while NN sensitivity is: $\frac{78}{78+1+1}=0.97$, meaning 97%. The data is combined in table 3.

Table 3: Comparison between NN and CNN

	NN		CNN	
Category	**sensitivity**	**precision**	**sensitivity**	**precision**
0	97%	86%	90%	100%
1	97%	99%	99%	99%
2	100%	97%	100%	96%
3	97%	99%	99%	99%
4	93%	90%	100%	98%
Average	**97%**	**97%**	**98%**	**99%**

The two classifiers has high percentage of precision and sensitivity, with slight advantage to CNN. Still, NN works faster and it makes it more suitable to our needs, so we selected using NN.

Testing procedure

The test was done when the vehicle was driving straight forward. During driving the video camera string is divided into individual frames. From each frame the lower 30% of the image is cut out. The rest of the frame is converted into grayscale image.

The resulted frame goes into the algorithm input. In case that the algorithm detects an area suspected to be a traffic sign, it generates an output vector with 4 fields cell element: the x and y coordinate values of the upper left end of the suspected area, the height and width of the suspected area. The number of vector's cells is as the number of the suspected areas detected in the image.

The information about the suspected areas is used to isolate each suspected area and convert it to a 28x28 pixels size. If the size of the suspected area is smaller than a predefined threshold, it is ignored since the identification of faraway traffic signs is not accurate.

The resulted image serves as an input to the identification algorithm. The output of the algorithm is a probability vector where each cell of the vector represents a specific traffic sign, with a probability that this traffic sign was identified in the related suspected area. If the probability is under 90%, the identification is ignored.

In addition correct identification will count only if a specific traffic sign was identified at least three times within a short period of 3 seconds. For that the algorithm used a counter for each sign type. The counter is automatically cleared after 3 seconds.

II. RESULTS

Now we will show the results from real world tests. Figure 17 presents No Entry traffic sign identification with tough environment conditions: darkness, light blurring and small image of the sign.

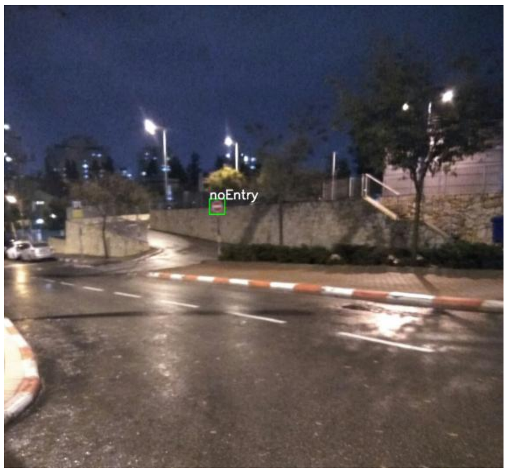


Fig. 17: No Entry traffic sign identification with tough environment conditions

Figure 18 presents give way traffic sign identification. Figure 19 presents Roundabout traffic sign identification.

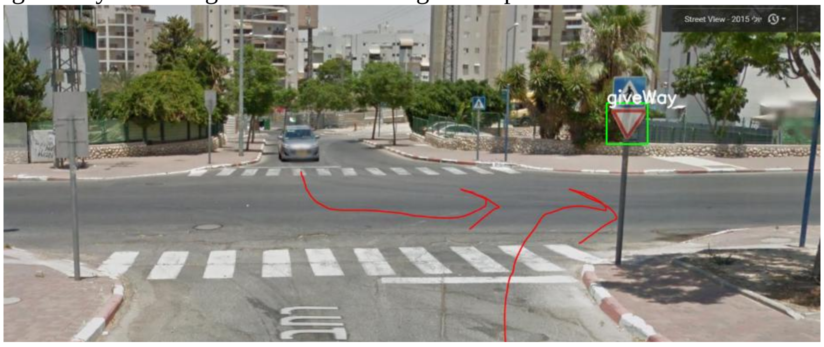


Fig. 18: Give way traffic sign identification

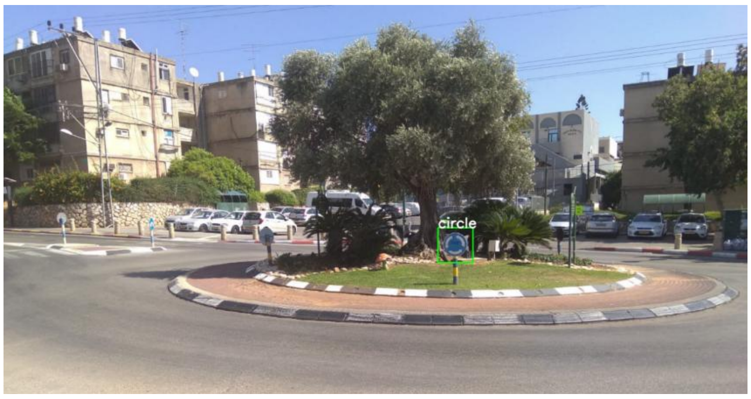


Fig. 19: Roundabout (Circle) traffic sign identification

Figure 20 presents successful identification of circle (roundabout) and give way. The other two false identifications were ignored. The right one with the limitation of 40 km/h was ignored since it had less than 90 percent probability to be a known traffic sign and the left one with the word go was ignored after 3 times check with the counter.

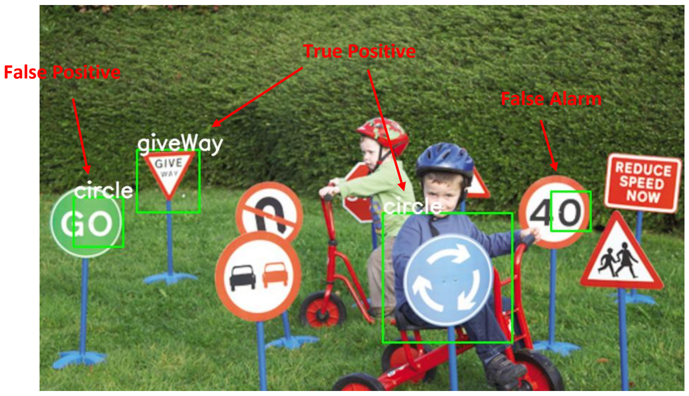


Fig. 20: Traffic signs identification

III. SUMMARY AND CONCLUSIONS

Traffic signs identification is one of the substantial challenges regarding autonomous vehicles driving. In this paper we presented several methods for traffic signs identification and we came to a conclusion that the most suitable methods are NN and CNN deep learning methods.
When comparing between NN and CNN, CNN presented better performances (99% of identification) but was too slow for real-time requirements. NN presented slightly less good performances (97% of performances) but was fast enough for real-time. This led us to the selecting NN as the preferred method.
The NN method worked as expected even with tough environment conditions.
The system can be improved by increasing the amount of traffic sign images for larger database.
The system can be expanded easily to take care about many more traffic signs. It will require adding data and re-train the system with all the traffic signs the existing ones and the new ones.
Further research might deal with improving CNN identification time. One option might be pre-processing for suspected areas and darkening the rest of the image.

BIBLIOGRAPHY

[1]. K. Bimbraw,"Autonomous Cars: Past, Present and Future - A Review of the Developments in the Last Century, the Present Scenario and the Expected Future of Autonomous Vehicle Technology", proc. of the 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2015), Colmar-Alsace, France, pp. 191-198, 2015.
[2]. B. Canis, " Issues in Autonomous Vehicle Testing and Deployment", Congressional Research Service, USA, 2020.
[3]. H. Fleyeh and M. Dougherty, "Road and Traffic Sign Detection and Recognition", Advanced OR and AI Methods in Transportation, 2006.
[4]. Y. Lai, N. Wang, Y. Yang and L. Lin, " Traffic Signs Recognition and Classification based on Deep Feature Learning", In Proceedings of the 7th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2018), pp. 622-629, 2018
[5]. L. Calderon,"Autonomous Cars – Safety and Traffic regulations", Robotics Tomorrow, 2018.
[6]. T. Keser, I. Dejanovic, "Traffic Sign Shape Detection and Classification based on the Segment Surface Occupancy Analysis and Correlation Comparisons", Tehnički vjesnik 25, Suppl. 1pp. 23-31, 2018.
[7]. A. Shustanov, and P. Yakimov, "CNN Design for Real-Time Traffic Sign Recognition", 3rd International Conference Information Technology and Nanotechnology, ITNT-2017, Samara, Russia, pp. 25-27, 2017.
[8]. S. Supriya, K. N. Rani and G. L. Raj, "Traffic road sign Detection and Recognition Using Geometric Shapes and Background Color: Laying a Foundation to Use Augmented Reality (A.R.) in Autonomous Vehicle Navigation and Decision Making", *Research Journal of Recent Sciences*, (5), pp. 17-20, 2016.
[9]. S. B. Wali , M. A. Abdullah, M. A. Hannan, A. Hussain, S. A. Samad , P. J. Ker and M. B. Mansor, "Vision-Based Traffic Sign Detection and Recognition Systems: Current Trends and Challenges", *Sensors*, 19, 2019.
[10]. E. Kolberg, "Traffic signs detection and identification using MOG and linear SVM methods", The International Journal of Engineering and Science (The IJES), 7 (12), 2018.
[11]. S. B. Wali, M. A. Hannan, A. Hussain, and S. A. Samad,"An Automatic Traffic Sign Detection and Recognition System Based on Colour Segmentation, Shape Matching, and SVM", Mathematical Problems in Engineering, 2015.
[12]. P. Viola and M. Jones, "Robust Real-time Object detection" International Journal of Computer Vision, 57(2), pp. 137-154, 2001.

E. Kolberg. "Traffic signs detection and identification using NN and CNN." *The International Journal of Engineering and Science (IJES)*, 9(7), (2020): pp. 07-19.