International Journal of Knowledge-based and Intelligent Engineering Systems 22 (2018) 205–214
DOI 10.3233/KES-180385
IOS Press

Recognition of traffic signs by convolutional neural nets for self-driving vehicles

Wael Farag[a,b]
[a]*College of Engineering and Technology, American University of the Middle East, Eqaila, Kuwait*
[b]*Electrical Power Engineering Department, Cairo University, Cairo, Egypt*
E-mail: wael.farag@aum.edu.kw, wael.farag@cu.edu.eg

Abstract. In this paper, a comprehensive Convolutional Neural Network (CNN) based classifier "WAF-LeNet" is proposed and developed to be used in traffic signs recognition and identification as an empowerment of autonomous driving technologies. The implemented architecture is a deep fifteen-layer network that has been selected after extensive trials to be fast enough to suit the designated application. The CNN got trained using Adam's optimization algorithm as a variant of the Stochastic Gradient Descent (SGD) technique. The learning process is carried out using the well-known "German Traffic Sign Dataset – GTSRB". The data has been partitioned into training, validation and testing data sets. Additionally, more random traffic signs images are collected from the web and further used to test the robustness of the proposed CNN classifier. The paper goes through the development process in details and shows the image processing pipeline harnessed in the development. The proposed approach proved successful in identifying correctly 96.5% of the testing data set and 100% of the robustness data set with much smaller and faster network than other counterparts.

Keywords: Deep learning, convolutional neural network, adam optimization, self-driving car, traffic sign recognition

1. Introduction

In the past decade, the automobile industry has made a shift towards intelligent vehicles equipped with driving assistance systems [1,2]. Recently the automobile industry has introduced vision systems in their high end cars. Also in the research community vision systems for traffic sign recognition are published [3–13]. All these systems are designed for their specific application as a pipeline of carefully tuned algorithms [3–6]. In the first step of the pipeline, the input is pre-processed with fixed algorithms such as lightning correction [5], histogram stretch [6], colour segmentation [3], edge detection [7], Hough transform [8], etc. The result of these steps is used to perform the classification by a matching algorithm or by machine leaning techniques such as Artificial Neural Networks (ANNs) [9–11] or Support Vector Machines (SVMs) [3]. Designing such a pipeline of algorithms is very time consuming and must be redone to support new road signs or other objects. The mapping of these computationally complex algorithms to on-board vision platforms is a time consuming task. This is especially true if the system is already deployed and the "vendor" wants to sell new functionalities to existing platforms.

In the last decade, deep learning has done a comprehensive progress and almost dominated the field of machine learning if compared to classical methods, especially after the recent advancements in GPUs (Graphics Processing Units). Accordingly, deep learning has produced astounding results in applications related to image recognition, natural language processing, autonomous cars, etc.

In this paper, our proposed solution differs from the current state-of-the-art systems in that it uses a platform independent flexible approach. The proposed vision system is based on a fully trainable CNN that is able to detect traffic signs in whatever shape; circular, rectangular or triangular. The CNN is trained for 43 different traffic signs but this number can be easily increased through transfer learning (keep adding

ISSN 1327-2314/18/$35.00

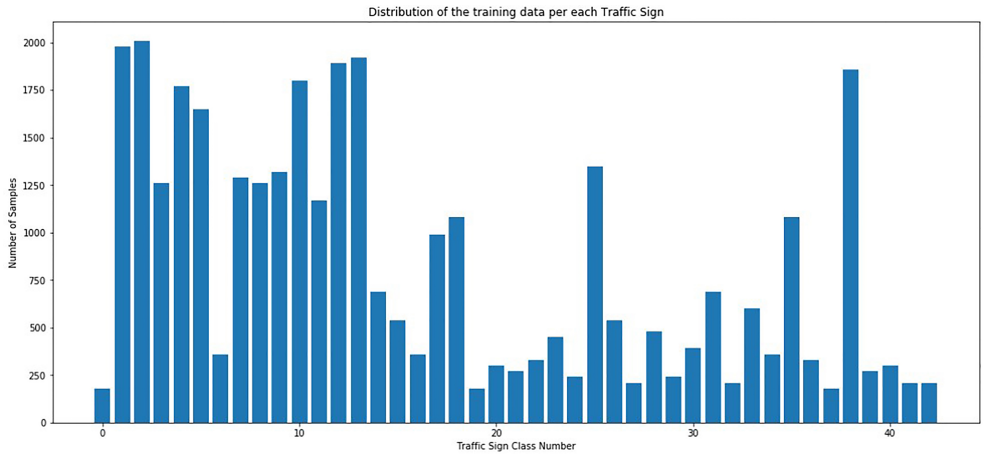

Fig. 1. The sharing of each traffic sign in the training data.

Fig. 2. Samples of training images from different classes.

new traffic signs training data and then use the current trained network after modifying the output layer). The inputs of the CNN are the raw pixel values and the outputs are direct confidence values representing the possibility of a specific traffic sign.

As an example for comparison purposes, the well-known work of Sermanet and LeCun [9] achieved a 99.17% accuracy, however, by using a much larger multi-scale convolutional neural network that is almost 100 times bigger than the one proposed in this paper. This is off-course makes its implementation in real time extremely costly or almost infeasible. Moreover, Qian et al. [12] proposed a CNN that has been trained as well on the same data set (GTSRB) to extract the visual attributes using what is called "Max Pooling Positions". However, the implemented CNN is almost 4 times the "WAF-LeNet" size (which means it is slower) and produces only 95.53% accuracy.

2. The training data set

The proposed CNN is being trained and validated using "The German Traffic Sign Recognition Benchmark" (GTSRB) [14]. The whole data set consists of 51,839 traffic sign image samples. The data is the partitioned into three partitions. The first partition is the training data which consists of 34,799 examples (67%). Each example is an image of size 32 $\times$ 32 pixels of RGB color. The training set has 43 different classes which means 43 different traffic sign. The data has not an even distribution of classes. For example, class 0 which is "20 km/h speed limit sign" has the lowest number of samples which is 180 sample image. Furthermore, class 2 which is "50 km/h speed limit sign" has the highest number of samples which is 2010 sample image. On brief, the data has an average of 809 sample per class. Figure 1 presents the sharing of each of the 43-class traffic signs in the training data set. Additionally, Fig. 2 shows examples of the images in the data set.

The second partition is the validation data set which consists of 4,410 examples (8.5%). Each example is an image of size 32 $\times$ 32 pixels of RGB color. The validation set has 43 different classes. Also, this data

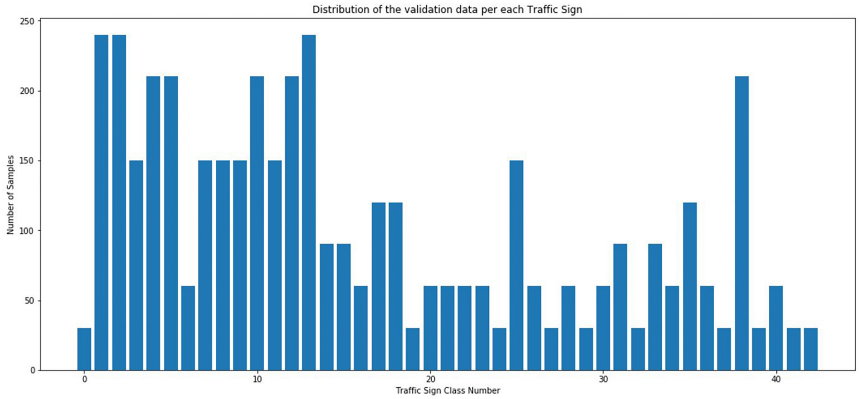

Fig. 3. The sharing of each traffic sign in the validation data.

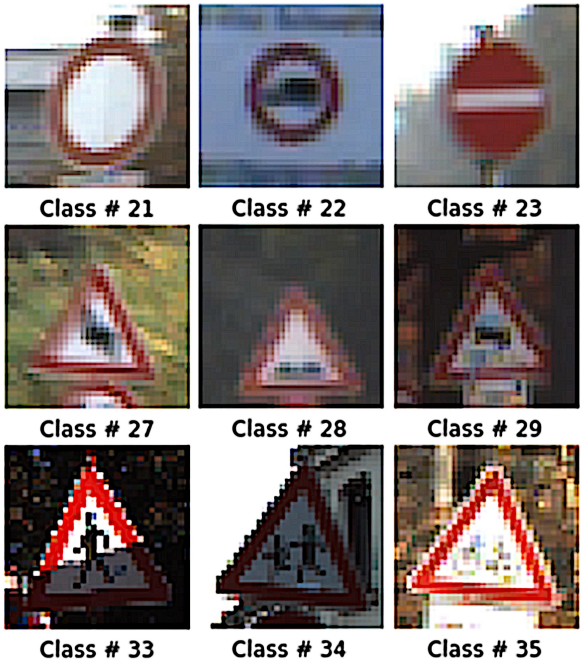

Fig. 4. Samples of training images from different classes.

has not an even distribution of classes. For example, class 0 which is "20 km/h speed limit sign" has the lowest number of samples which is 30 sample image, and class 1 which is "30 km/h speed limit sign" has the highest number of samples which is 240 sample image. Briefly, the data has an average of 102 samples per class. Figure 3 presents the sharing of each of the 43-class traffic signs in the validation data set. Moreover, Fig. 4 shows examples of the images in the data set.

The third partition is the test data set which consists of 12,630 examples (24.4%). Each example is an image of size 32 $\times$ 32 pixels of RGB color. The test set has 43 different classes and the data has not an even distribution of classes. For example, class 0 which is "20 km/h speed limit sign" has the lowest number of samples which is 60 sample images, while class 1 which is "30 km/h speed limit sign" has the highest number of samples which is 750 sample images. Consequently, the data has an average of 293 samples per class. Figure 5 presents the sharing of each of the 43-class traffic signs in the test data set, while Fig. 6 shows examples of the images in the data set.

3. Traffic images pre-processing

Before using the training/validation images, these images need to be pre-processed to make more useful and convenient throughout the learning process. The pre-processing steps meant to improve the training results and reduce the computation as much as possible. The following steps describes the implemented pre-processing steps in order of execution:

I. Normalization (colour): for colour images by simply implementing a min-max scaling for colour image data. Each RGB colour gets scaled down to 0.1 $\rightarrow$ 0.9 range instead of the 0 $\rightarrow$ 255 range.

II. Converting the colour training images to grey: this is done using the OpenCV [15] function "ctvColor". This step reduces the training computation to 1/3rd its previous estimate as now each pixel is represented with one pixel instead of 3.

III. Normalization (grey): this is done for grey images by simply implementing a min-max scaling (between 0.1 and 0.9) for grey image data.

IV. Filtering the noise: this is done using the OpenCV [15] function "GaussianBlur" which

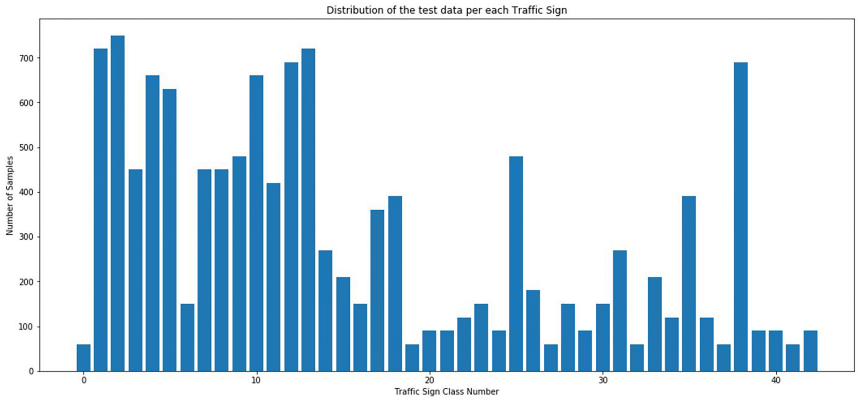

Fig. 5. The sharing of each traffic sign in the test data.

executes the Gaussian filter algorithm, with a kernel size of 5.

V. Shuffling the training data: this is done once each training epoch to avoid pattern memorization and consequently trapping in local minima.

All the above steps as been tried. Pre-processing using colour images has been tried. Also pre-processing using Gaussian blur filtering is also tried. We found out that using grey images and "no blur filtering" gives the best performance as per our many trial-and-error endeavours (which supports the results in [9]).

Finally, the actually used pre-processing pipeline is: Colour-Image $\rightarrow$ Convert-to-Grey $\rightarrow$ Normalization $\rightarrow$ Shuffling.

4. The convolution neural network

The proposed convolution network consists of basically four main types of layers: the convolutional, the ReLU layer, the polling layer, and the fully connected layer [9,16].

1. The convolutional layer: The big pixel matrix (digital image) will be divided into over-lapped sub-matrixes (feature maps) by sliding a kernel or feature detector (filter) over the original images and applying a dot product between them. The detected features are depending on the values of the filter. The size of the feature maps is depending on the numbers of filters (depth), the number shifting steps of the filter (stride) and the existence of the zero-padding (padding the feature map with zeros at the bounders).
2. The ReLU: The Rectified Linear Unit (ReLU) is a non-linearity operation that applies directly after the convolution to change the negative values for each pixel by a zero. The purpose of ReLU is to introduce non-linearity in our CNN, since most of the real-world data as well as traffic signs data that the CNN needs to learn would be non-linear.
3. The pooling layer: Polling is a nonlinear down sampling that uses squared filters (such 2×2 filter) to reduce noise, keep the interesting features of the image, makes the process faster and allows the next layers to analyse large parts of the image without increasing the filter size. Spatial pooling reduces the dimensionality of each feature map but retains the most impor-

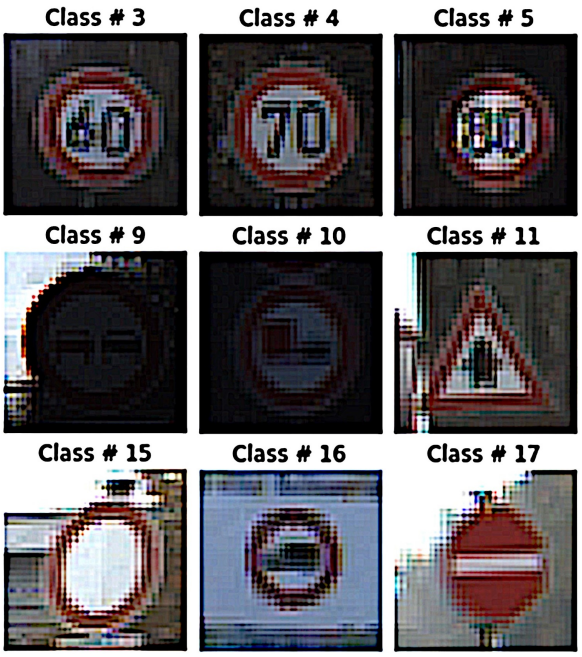

Fig. 6. Samples of training images from different classes.

Table 1
WAF-LeNet architecture

#	Layer	Size/Output	Parameters	Comment
1	Input	$32 \times 32 \times 1$	–	Grey scale
2	Conv. #1	$28 \times 28 \times 6$	Kernel: 5×5, Strides: 1×1, Padding: Valid	With bias (6)
3	Activation #1	$28 \times 28 \times 6 = 4704$	ReLU	
4	Pooling #1	$14 \times 14 \times 6$	Max, K: 2×2, Strides: 2×2, Padding: Valid	
5	Conv. #2	$10 \times 10 \times 16$	Kernel: 5×5, Strides: 1×1, Padding: Valid	With bias (16)
6	Activation #2	$10 \times 10 \times 16 = 1600$	ReLU	
7	Pooling #2	$5 \times 5 \times 16$	Average, K: 2×2, Strides: 2×2, Padding: Valid	
8	Flat	400	–	Just flatten the input
9	Fully connected #1	300		With bias (300)
10	Activation #3	300	ReLU	
11	Drop-out	300		With bias (300)
12	Fully connected #2	150		With bias (150)
13	Activation #4	150	ReLU	
14	Drop-out	150		Keep probability: 0.75
15	Fully connected #3	43	One-hot (Softmax)	The number of Traffic Sign Classes

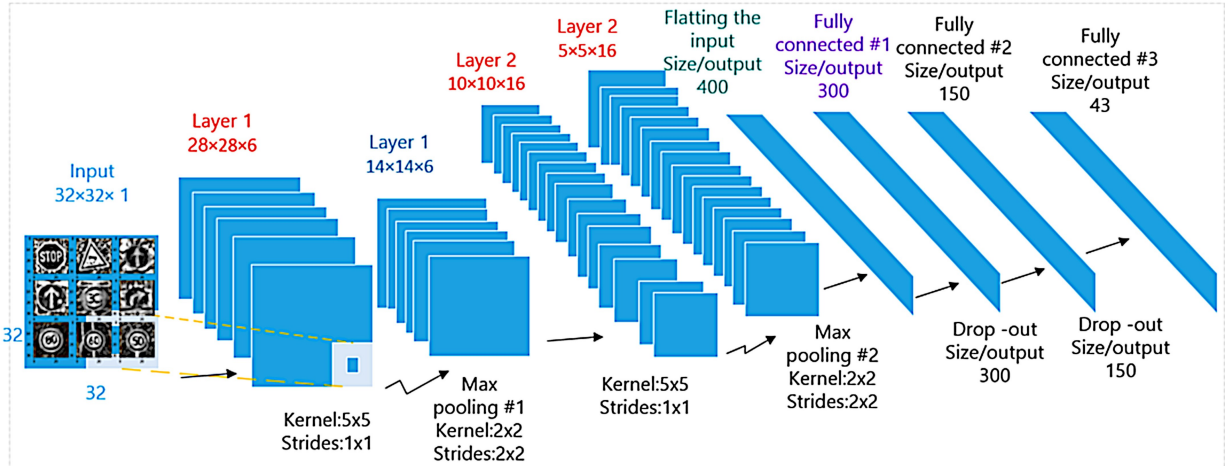

Fig. 7. WAF-LeNet detailed structure.

tant information. Spatial pooling can be of different types: Max, Average, Sum, etc.

4. The fully connected layer: The fully connected layer is concerned of classification and decision making after the back propagation. It calculates the percentage of error then repeats the previous three steps and change the weights until reach the minimum percentage of error. This layer is a traditional Multi-Layer Perceptron that uses a Softmax activation function in the output layer. The term "Fully Connected" implies that every neuron in the previous layer is connected to every neuron on the next layer.

5. The CNN model architecture

The implemented neural network model is an upgraded version of LeNet architecture [17] and given the name WAF-LeNet. Figure 7 depicts the detailed structure of WA-LeNet as well as Table 1 presents the layers specific information.

Two drop-out layers are added and the 1st and 2nd fully connected layers are widened. Also, average Pooling are used instead of the Max-Pooling is layer 7.

6. The learning algorithm

In this work, Adam learning algorithm [18] is used to train the proposed WAF-LeNet and update its weights iteratively based on the GTSRB training data [14]. Adam is an optimization algorithm that is used instead of the classical Stochastic Gradient Descent (SGD) learning algorithm [19]. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.

Adaptive Moment Estimation (Adam) [18] is a comprehensive technique that calculates the adaptive learning rates for each parameter in the neural network individually. Moreover, an exponentially decaying average of past squared gradients v_t, is stored for further processing. Adam technique keeps as well an exponentially decaying average of past gradients m_t. Both v_t and m_t are calculated as follows:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{aligned} \tag{1}$$

m_t and v_t are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients g_t, respectively, which the name of the method came from. If m_t and v_t are initialized as vectors of 0's, it was observed that they are biased towards zero, especially during the initial time steps, and especially when the decay rates are small (i.e. β_1 and β_2 are close to 1).

These biases have been counteracted by calculating instead bias-corrected first and second moment estimates:

$$\begin{aligned} \widehat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \widehat{v}_t &= \frac{v_t}{1-\beta_2^t} \end{aligned} \tag{2}$$

Then, the above estimates ($\widehat{m}_t$, $\widehat{v}_t$) are used to update the parameters which yields the principal Adam update rule:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{v}_t} + \epsilon} \widehat{m}_t \tag{3}$$

for all neural network model's parameters $\theta \in \mathbb{R}^d$ (weights and biases), where η is the learning rate and ϵ is a very small constant prevents divide by zero.

In the main paper that presents the Adam technique [20], it was proposed to use the default values of 0.9 for β_1, 0.999 for β_2, and 10^{-8} for ϵ. It has been empirically shown as well that Adam works well in both experimental and actual applications and is favorable if compared to other adaptive learning techniques [20].

The output layer of the proposed WAF-LeNet is a softmax function. The softmax function or normalized exponential function [21] is a generalization of the logistic function that "squashes" a K-dimensional vector "z" of arbitrary real values to a K-dimensional vector $\sigma(z)$ of real values, where each entry is in the range (0, 1), and all the entries add up to 1. The function is given by

$$\sigma(z) : \mathbb{R}^k \rightarrow \left\{ \sigma \in \mathbb{R}^k | \sigma_i > 0, \sum_{i=1}^{K} \sigma_i = 1 \right\} \tag{4}$$

$$\sigma(z)_i = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, 2, \ldots, K. \tag{5}$$

In probability theory, the output of the softmax function can be used to represent a categorical distribution – that is, a probability distribution over K different possible outcomes. Specifically, in multinomial logistic regression, the input to the function is the result of K distinct linear functions, and the predicted probability for the j'^{th} class given a sample vector x and a weighting vector w is:

$$P(y=i|x) = \frac{e^{x^T w_j}}{\sum_{k=1}^{K} e^{x^T w_k}} \quad \text{for } j = 1, 2, \ldots, K. \tag{6}$$

The *data loss*, which in a supervised learning problem measures the compatibility between a prediction (e.g. the class scores in classification) and the ground truth label. The data loss takes the form of an average over the data losses for every individual example. That is, $L = \frac{1}{N} \sum_i L_i$, where N is the number of training data. Let's abbreviate $f = f(x_i; W)$ to be the activations of the output layer in the CNN. The cross-entropy loss for a single training sample with a ground truth label y_i (for the softmax classifier) is given by:

$$L_i = -\log \left(\frac{e^{f_{y_i}}}{\sum_j e^{f_i}} \right) \tag{7}$$

Moreover, the multi-class cross entropy can them be formulated as follows [21–23]:

$$\begin{aligned} L_i = \sum_j y_{ij} \ & \log(\sigma(f_i)) + \\ & (1 - y_{ij}) \log(1 - \sigma(f_i)) \end{aligned} \tag{8}$$

where the labels y_{ij} are assumed to be either 1 (positive) or 0 (negative) and $\sigma(\cdot)$ is the sigmoid function. Furthermore, the gradient of L_i w.r.t. f_i is simply give as: $\frac{\partial L_i}{\partial f_j} = y_{ij} - \sigma(f_i)$.

7. The model training

The WAF-LeNet model is trained using the parameters listed in Tables 2–4 using Adam's optimization algorithm. The training results are presented in Table 5.

The training of the WAF-LeNet has been carried-out through several trials to achieve the presented results. The following observations has been composed during the training process:

1. Several trials have been carried out to train the current WAF-LeNet architecture using the RGB images after being normalized, however, the results were not good enough, and the validation accuracy didn't cross the 91% level.

Table 2
WAF-LeNet training parameters (learning rate)

Parameter	Value	Comment
Learning rate η	0.001	For epochs: 000–100
	0.0005	For epochs: 101–125
	0.0003	For epochs: 126–150
	0.0002	For epochs: 151–175
	0.0001	For epochs: 176–200

Table 3
WAF-LeNet training parameters (PKeep)

Parameter	Value	Comment
Drop-out layers keep probability (PKeep)	0.75	For epochs: 000–100
	0.80	For epochs: 101–125
	0.85	For epochs: 126–150
	0.90	For epochs: 151–175
	1.00	For epochs: 176–200

Table 4
WAF-LeNet training parameters (others)

Parameter	Value	Comment
Batch size	128	For epochs: 000–200
Epochs	200	The whole training

Table 5
WAF-LeNet training results

Parameter	Value	Comment
Training-data accuracy	100.0%	Highest reached
Validation-data accuracy	96.4%	Highest reached
Testing-data accuracy	94.5%	Highest reached

2. Then the training trials have been switched to use the normalized converted grey-image data with more successful results with the validation accuracy reached 93% but didn't go much further beyond that.
3. Using Gaussian filtering for the both Grey and RGB images didn't improve the accuracy, in contrary, it worsen the accuracy much further. For this reason, this technique has been skipped.
4. In order to improve the performance further, two drop-out layers have been added as shown in Table 1 as well as increasing the breadth of the 1st and 2nd fully connected layers. This approach has significantly improved the results and the validation accuracy crossed the 95% mark.
5. For further improvement, the learning rate as well as the keep probability of the drop-out layers have been updated according to the profiles shown in Tables 1 and 2. This approach has significantly improved the validation accuracy to reach 96.4% as shown in Table 5.
6. The WAF-LeNet after training has been applied on the testing data resulting in 94.5% accuracy.

Fig. 8. Raw traffic sign samples from the web.

8. The robustness testing

Ten new random traffic sign images have been downloaded from the web for further robustness testing. Samples of these raw images are shown in Fig. 8.

The fully trained WAF-LeNet has been tested (as illustrated in Fig. 9) on these 10 raw images (after being resized to 32 × 32 using OpenCV resize function) with very low results achieved (only 50%). However, after implementing some processing on the images mainly cropping, centering, and rotation as shown in Fig. 10, then the images have been resized to 32 × 32 and fed to the network with 100% results in testing accuracy achieved.

To have more insight into the performance of the proposed WAF-LeNet, Fig. 11 shows how the classifier identified the traffic sign by showing the generated internal top five probabilities (the Softmax) in "layer 15" before applying the one-hot concept (taking the maximum as the final output). The graph shows that the classifier identified the traffic sign image on the left side as "ID#1 – 30 km/h speed limit" with 88% probability and as "ID#0 – 20 km/h speed limit" with 10% probability and as "ID#38 – keep right" with 2% probability. This examples shows how robust the proposed classifier is. For more insight how the CNN works, Fig. 12 depicts the 1st convolutional layer "layer 2" output feature map.

Furthermore, Fig. 13 shows the generated internal top five probabilities in "layer 15" with the traffic sign "ID#18 general caution" identified with a probability of almost 100%. Moreover, for additional elaboration the feature map of the 1st convolutional layer is depicted in Fig. 14.

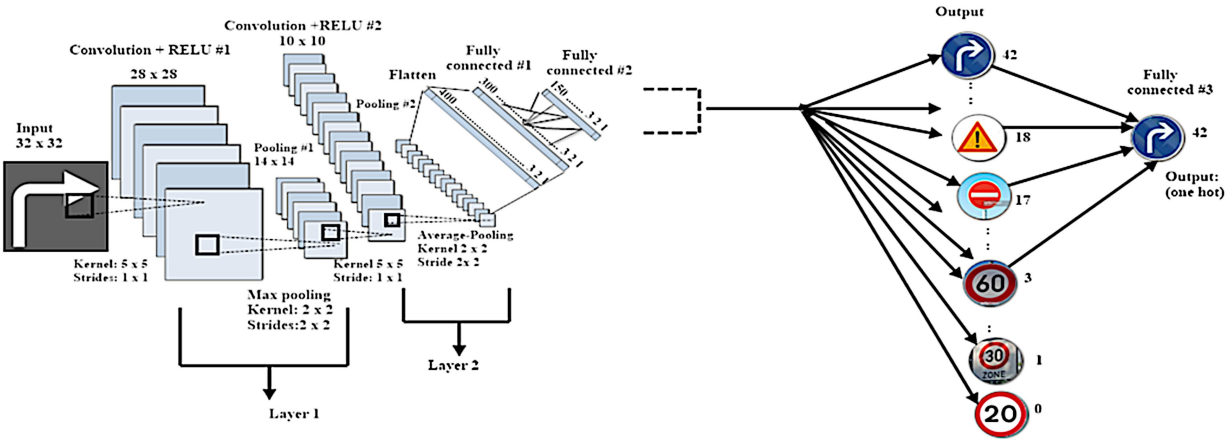

Fig. 9. Illustration of one-hot activation.

Fig. 10. Cropped and centred traffic web sign samples.

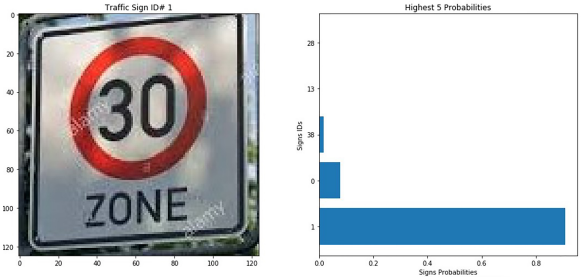

Fig. 11. Internal generated probabilities for different classes.

9. Shortcoming of the implemented approaches

The presented results and the further analysis identifies some shortcomings that are listed below:

1. Further investigation on the use the RGB images instead of grey ones needs to be carried out. Even though using grayscale images reduces complexity and computation but it deprives the CNN from precious information (like colours) is believed to be very crucial in classifying traffic signs.
2. Given the criticality of the traffic sign recognition for the success of autonomous driving, much more training data is required to improve the performance. This is specifically true for classes that have relatively low number of samples like Classes 0, 41 and 42.
3. There is a very big difference between classes in terms of the number of available samples in the training data which reflects in the tendency of recognizing the classes with larger number of samples better. Therefore, more work is needed to be done to narrow this gap.

Based on the results achieved on the original testing data set of 94.5% (Table 5), if the tested images are well-centered and well-cropped (which means the traffic sign fills 75% of the image or more) as shown in Figs 2 and 10, the WAF-LeNet in its current state works with good performance. Otherwise, the network performs insufficiently.

10. Suggested improvements

Based on the achieved results and the above findings, the below points summarize the suggested improvements:

1. Increasing of the depth of the CNN (adding more convolutional layers and feature-extraction filters) and the using of RGB images instead of grey scale ones should improve the model per-

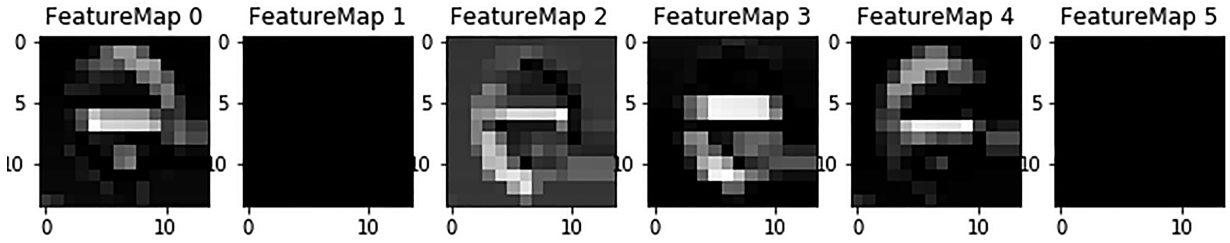

Fig. 12. 1st convolutional layer output feature map.

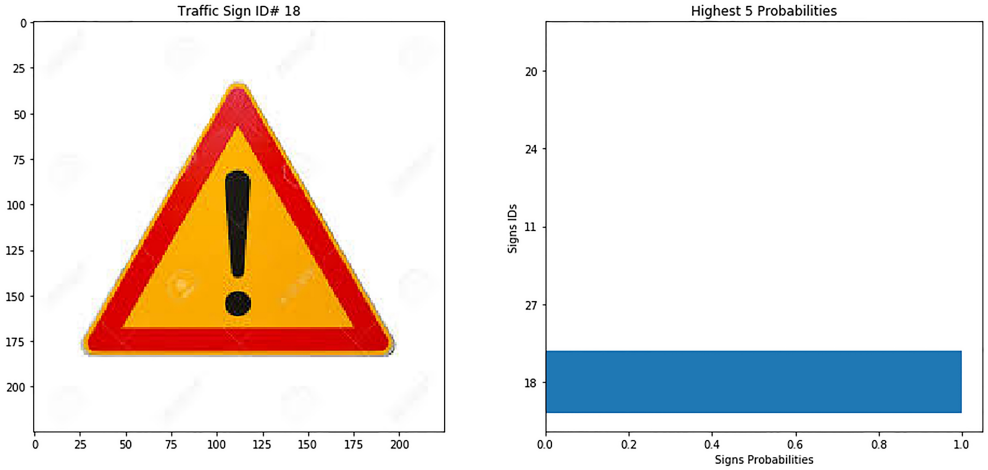

Fig. 13. Internal generated probabilities for different classes.

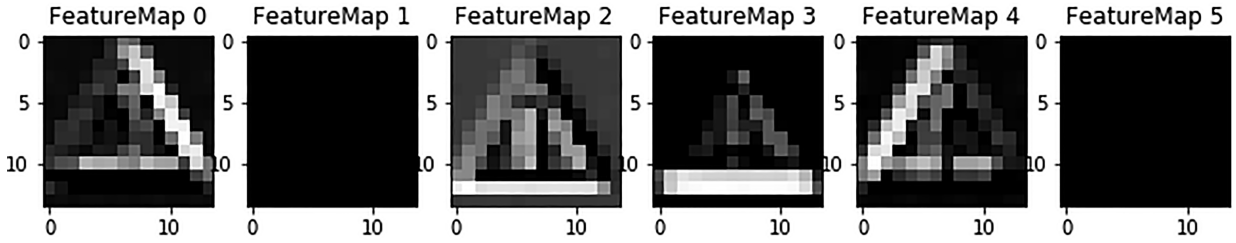

Fig. 14. 1st convolutional layer output feature map.

formance significantly, as converting images to grey makes the data loose vital information about the traffic signs. Therefore, more trials need to be carried out using colour images with deeper and wider networks.

2. Equalizing the number of samples in each class by generating more data from the available data by the process of data augmentation (skewing, rotation, noise addition, etc.).
3. Data augmentation is believed to be a good solution to size up the training samples. It can be done by applying some image processing on the current data samples like skewing, flipping, adding random bright and dark spots, adding dusty noise, etc. This should significantly improve the classification robustness.
4. Adding "Skip Connections" to the CNN (connections that skip layers) to extract the "global" and invariant shapes and structures [9] of the traffic signs may considerably improve the performance.
5. The WAF-LeNet is mainly used for recognition. In order to complete the whole functionality, it should be augmented with another model/network that focuses on detection and localization of the traffic sign within the big image [24,25].

11. Conclusion

In this paper, a CNN-based classifier "WAF-LeNet" has been proposed. The architecture of the CNN is presented in details. The structure of the comprehensive training, validation and testing data is described. The involved image processing algorithms have been described as well and their contributions are analysed. WAF-LeNet has shown a very good performance in recognizing 43 different traffic signs classes. Its accuracy reached 96.4% and its robustness test achieved 100% of correct identifications. The shortcomings of the proposed approach have been discussed with proposed improvement actions for future work being elaborated. The presented solution presents a corner stone in facilitating the existence of full autonomous cars in the near future.

References

[1] K. Mansour and W. Farag, AiroDiag: A sophisticated tool that diagnoses and updates vehicles software over air, in: *the 2012 IEEE Intern Electric Vehicle Conference (IEVC)*, TD Convention Center Greenville, SC, USA, March 4, 2012, ISBN: 978-1-4673-1562-3.

[2] W. Farag, CANTrack: Enhancing automotive CAN bus security using intuitive encryption algorithms, in: *7th Inter Conf on Modeling, Simulation, and Applied Optimization (ICMSAO)*, UAE, March 2017.

[3] G. Kiran, L.V. Prabhu, V. Abdu-Rahiman and K. Rajeev, Traffic sign detection and pattern recognition using support vector machine, in: *7th Inter Conf on Advances in Pattern Recognition, ICAPR'09*, Kolkata, India, 2009.

[4] L. Chen, Q. Li, M. Li and Q. Mao, Traffic sign detection and recognition for intelligent vehicle, in: *IEEE Intelligent Vehicles Symposium*, Baden-Baden, Germany, June 2011.

[5] Z.T. Kardkovács, Z. Paróczi, E. Varga, A. Siegler and P. Lucz, Real-time traffic sign recognition system, in: *2nd Inter Conf on Cognitive Infocommunications (CogInfoCom)*, Budapest, Hungary, August 2011.

[6] J. Greenhalgh and M. Mirmehdi, Real-time detection and recognition of road traffic signs, *IEEE Trans on Intelligent Transportation Systems* **13**(4) Dec (2012).

[7] G. Loy, Fast shape-based road sign detection for a driver assistance system, in: *Proc IROS*, 2004, pp. 70–75.

[8] P. Yakimov and V. Fursov, Traffic signs detection and tracking using modified hough transform, in: *Proc of 12th ICETE*, Colmar, France, 2015.

[9] P. Sermanet and Y. LeCun, Traffic sign recognition with multi-scale convolutional networks, in: *The 2011 International Joint Conference on Neural Networks (IJCNN)*, San Jose, CA, USA, DOI: 10.1109/IJCNN.2011.6033589.

[10] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li and S. Hu, Traffic-sign detection and classification in the wild, in: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, June 2016.

[11] 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* **99** (2018), 158–165, Elsevier.

[12] R.Q. Qian, Y. Yue, F. Coenen and B.-L. Zhang, Traffic sign recognition using visual attribute learning and convolutional neural network, in: *Inter Conf on Machine Learning and Cybernetics (ICMLC)*, Jeju, South Korea, July 2016.

[13] W. Farag and Z. Saleh, Traffic signs identification by deep learning for autonomous driving, in: *Smart Cities Symposium (SCS'18)*, Bahrain, 22–23 April, 2018.

[14] J. Stallkampa, M. Schlipsing, J. Salmena and C. Igel, Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition, *Neural Networks* **32** (2012), 323–332, Elsevier.

[15] OpenCV Python Library, https://opencv.org/.

[16] U. Karn, An intuitive explanation of convolutional neural networks, https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/.

[17] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, Gradient-based learning applied to document recognition, *Proceedings of the IEEE* **86**(11) November (1998), 2278–2324.

[18] D.P. Kingma and J. Ba, Adam: A method for stochastic optimization, in: *3rd Inter Conf for Learning Representations*, San Diego, USA, 2015.

[19] L. Bottou, Online algorithms and stochastic approximations, in: *Online Learning and Neural Nets*, Cambridge Univ Press, 1998, ISBN: 978-0-521-65263-6.

[20] S. Ruder, An overview of gradient descent optimization algorithms, arXiv:1609.04747v2, 15 Jun 2017.

[21] M. Christopher, *Bishop, Pattern Recognition and Machine Learning*, Springer, 2006.

[22] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2016.

[23] K. Murphy, *Machine Learning: A Probabilistic Perspective*, MIT, 2012, ISBN: 978-0262018029.

[24] X.Z. Han, C. Chen, K.J. Wang, Y. Yuan and Y. Song, Study on detection and localization algorithm of traffic signs from natural scenes, *Advances in Mechanical Engineering*, February 2015.

[25] A. Møgelmose, M.M. Trivedi and T.B. Moeslund, Traffic sign detection and analysis: Recent studies and emerging trends, in: *15th Inter IEEE Conf on Intelligent Transportation Systems*, 2012, pp. 1310–1314, DOI: 10.1109/ITSC.2012.6338900.

[26] M. Nagiub and W. Farag, Automatic selection of compiler options using genetic techniques for embedded software design, in: *IEEE 14th Inter Symposium on Comp Intelligence and Informatics (CINTI)*, Budapest, Hungary, Nov 19, 2013, ISBN: 978-1-4799-0194-4.