

ISTANBUL AYDIN UNIVERSITY

PROJECT PAPER

**AI TRAFFIC SIGNS RECOGNITION**

ABDEL AZIZ KAMO MEGNA (Y2013.140009)

Supervisor: Dr HALUK GUMUSKAYA

21th January, 2021

Abstract

As the number of innovations in artificial intelligence increases at an exponential rate, a recurring problem is that concerning the traffic, the driver is a human and he may miss some of the traffic signs on the road, so the car has to be able to correct this error. For a vehicle to be fully autonomous, it must be able to recognize and identify in real time the traffic signs on the road. In this project, an autonomous system for the detection and identification of traffic signs using artificial intelligence is implemented to try to make vehicles much more autonomous and to further develop the artificial intelligence sector.

Keywords—Classification, Recognition, Artificial Intelligence, Convolutional Neural Network (CNN), Multilayer Perceptron (MLP), Deep learning, Artificial Intelligence, Road signs, Autonomous vehicles.

1. Introduction

In current traffic management systems, there is a high probability that the driver may miss some of the traffic signs on the road because of overcrowding due to neighbouring vehicles. With the continuous growth of vehicle numbers in urban agglomerations around the world, this problem is only expected to grow worse.

A visual-based traffic sign recognition system can be implemented on the automobile with an aim of detecting and recognizing all emerging traffic signs. The same would be displayed to the driver with alarm-triggering features if the driver refuses to follow the traffic signs. Traffic Sign Recognition (TSR) is used to display the speed limit signs. Here, OpenCV is used for image processing. OpenCV is an Open-source Computer Vision library designed for computational efficiency with a strong focus on real time applications. So in this project we will train and classify Traffic Signs using Convolutional neural networks. This will be done using OPENCV in real time using a simple webcam.

1. Related Work

Nowadays, there is a growth evolution in the development of intelligent transportation systems (ITS) Self-Driving Cars (SDC). In these systems, traffic signs detection and recognition is one of the difficult tasks that confront researchers and developers. This issue is addressed as a problem of detecting, recognizing, and classifying objects (traffic signs) using computer vision and still be a challenge until now. The work presented in this paper focuses on traffic signs recognition without the consideration of the detection step. For this purpose, this section discusses only related works from this angle. Traffic signs recognition is divided in two parts: features extraction and sings recognition. In the first step, several methods have been proposed, including edge detection [1], scale invariance feature (SIFT) [2], speeded-up robust feature (SURF) [3], Histogram of gradient (HOG) [4] and others. In [5], Bag of Words (BOW) exploiting SURF and k-means classifier was used. Typically, the output of this step is the input of the classification algorithms for the recognition of the road signs. Many algorithms have been used such as K-Nearest Neighbor (KNN) classifier [3], Support Vector Machine (SVM) [6] and neural network [5][7] for traffic signs classification. Authors in [5] proposed the evaluation of three methods namely, Artificial Neural Network (ANN), Support Vector Machine (SVM) and Ensemble subSpace KNN using BoW where every road sign is encoded with 200 features. The Multi-layer Perceptron Neural network provides better results. Currently, Convolutional networks are gradually replaced traditional computer vision algorithms for different applications such as object classification and pattern recognition [7][8]. It is used for the extraction and the learning of depth description of the traffic signs. This solution overcomes the step of descriptors extraction which is very sensitive to different factors. This network takes 2D image and processes it with convolution operations. It has the ability to learn a representative description of image

1. **Overview, Methods and Tools**

In this project, we are going to train and classify traffic signs using Convolutional Neural Networks (CNN), it will be done using Opencv in real-time using our webcam, and in this project we will train traffic science with over 35000 images of 43 differents clases whith the help of tensorflow and keras.

In this case of image processing using CNN is the right approach because it is one of the most popular neural network architectures. CNNs are extremely successful at image processing, but also for many other tasks (such as speech recognition, natural language processing, and more). The state of the art CNNs are pretty deep (dozens of layers at least), so they are part of Deep Learning. But we can build a shallow CNN for a simple task, in which case it's not (really) Deep Learning.

But CNNs are not alone, there are many other neural network architectures out there, including Recurrent Neural Networks (RNN), Autoencoders, Transformers, Deep Belief Nets (DBN = a stack of Restricted Boltzmann Machines, RBM), and more. They can be shallow or deep, even thus shallow RNNs can be considered part of Deep Learning since training them requires unrolling them through time, resulting in a deep net.

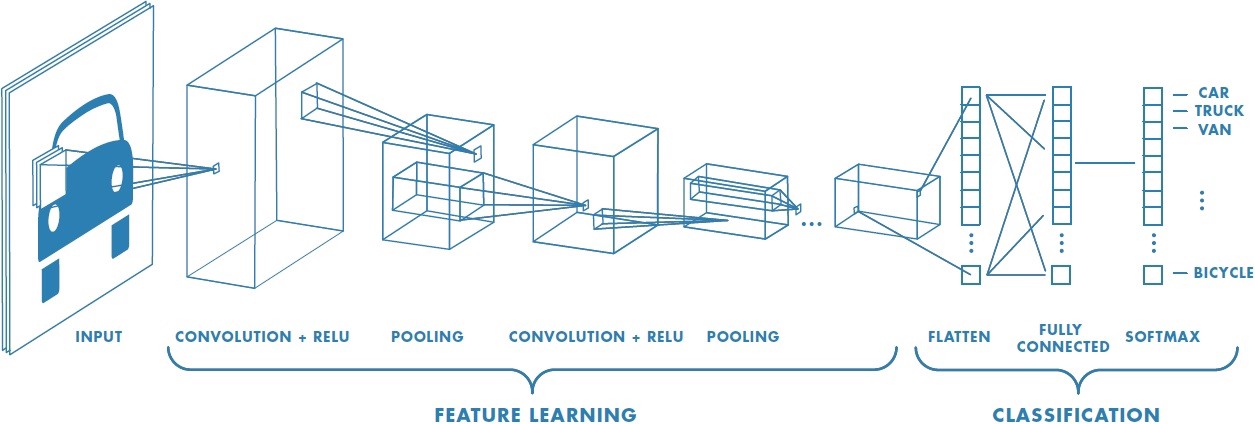


Figure1: understanding of Convolutional Neural Network (CNN)

In this project we used **OpenCV** which is a cross-platform library with that we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.[14]

For the build and train of our model, we used **Keras** and **Tensorflow**. Keras is a neural network library while TensorFlow is the open-source library for a number of various tasks in machine learning. TensorFlow provides both high-level and low-level APIs while Keras provides only high-level APIs. Both frameworks thus provide high-level APIs for building and training models with ease.[15]

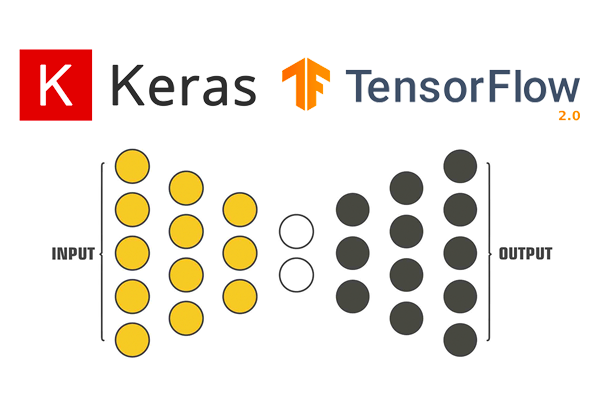


Figure:2 The use of Keras and Tensorflow

In this image we can clearly see that the helps of keras is to provide the input while the helps of tensorflow is to provide the output result.

1. **Architecture, Algorithms, Models and Data**

* **The Neural Network Architecture**

Using a fully connected neural network to make an image classification requires a large number of layers and neurons in the network, which increases the number of parameters leading the network to over-fitting (memorizing the training data only). The input image may also lose its pixels correlation properties since all neurons (carrying pixels values) are connected to each other [7]. Convolutional neural networks have emerged to solve these problems through their kernel filters to extract main features of the input image and then inject them into a fully connected network to define the class [7].

* **Traffic signs dataset**

A rich dataset is needed in object recognition based on neural network in order to train the system and evaluate its results. For the purpose of traffic signs classification, we used the German Traffic Sign Recognition Benchmark (GTSRB) dataset, which contains over than 35000 of images divided into 43 different classes.

|  |  |  |  |
| --- | --- | --- | --- |
| Category | Task | Percentage of images | Shape |
| Training data | Used to train the network | 70% | 4 dimensions tensor to determine the image index in the dataset, the pixel’s row-column and the information it carries (Red Green Blue value) |
| Validation data | Allows to supervise the network performances while training it (a reduced version of testing data) | 20% |
| Testing data | Used to evaluate the final network | 10% |

Table1: dataset repartition

* **Training and testing**

The unbalanced distribution of images in the German Traffic Sign Benchmark privileges some classes over others during the training phase because they are better represented in terms of number of images. In order to make sure that the learning of the network is well performed, a data augmentation of some classes is done by applying some geometric transformations (rotation, translation, and shear mapping) on many of their images.

We split our data in training and testing phase as follow: 70 percent for the training, 10 percent for the testing and 20 percent for the validation. We done our training data after create our convolutional neural network, using tensorflow we train it ad print the result. Training our model took about 60 minutes because we set the variable epoch\_val to 10 and each epoch was about 6.1 minutes.

To improve your training time, we use three callbacks such as: **tf.keras.callbacks.EarlyStopping, tf.keras.callbacks.ModelCheckpoint and tf.keras.callbacks.TensorBoard**

1. **Experiments, Analysis and Performance**

**5.1. Building the Model, Training and the Problems Faced**

It is important to know that we do not have the same number of images for each class

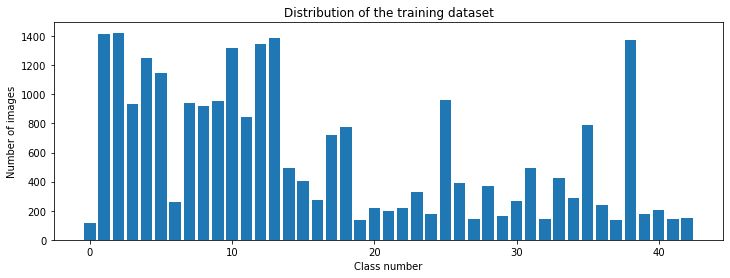


Figure3: Distribution of the training dataset

So here we can see we have about 100 image for the first class, and then we have about 13000 images for the second class, so the distribution is not even. We might get good classification for one class and bad classification for other because the dataset does not have the same number of image for each class. So we can learn by this project how much dataset is required to have a good classification.

The first way we use to train our model was without callbacks and then I use callbacks to improve our training time.

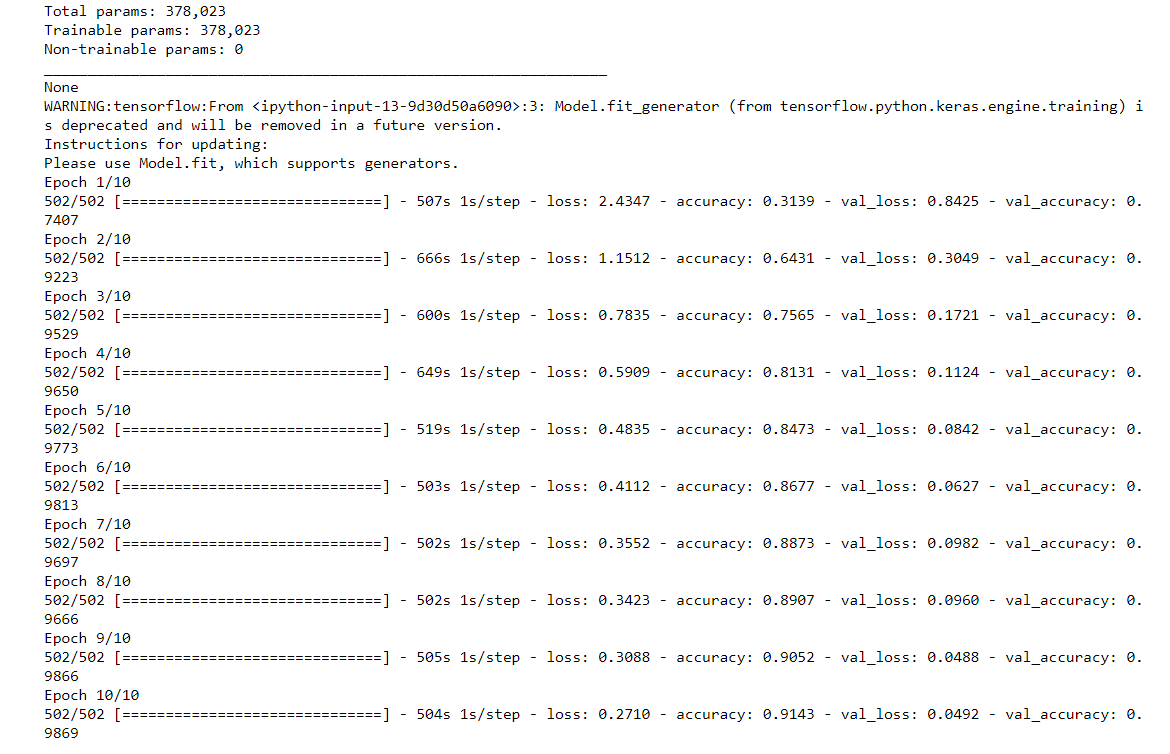


Figure4: training model without callbacks

In this figure we can see that the training time for each epochs is on average 610s and our total processing time is about 1 hour or more.

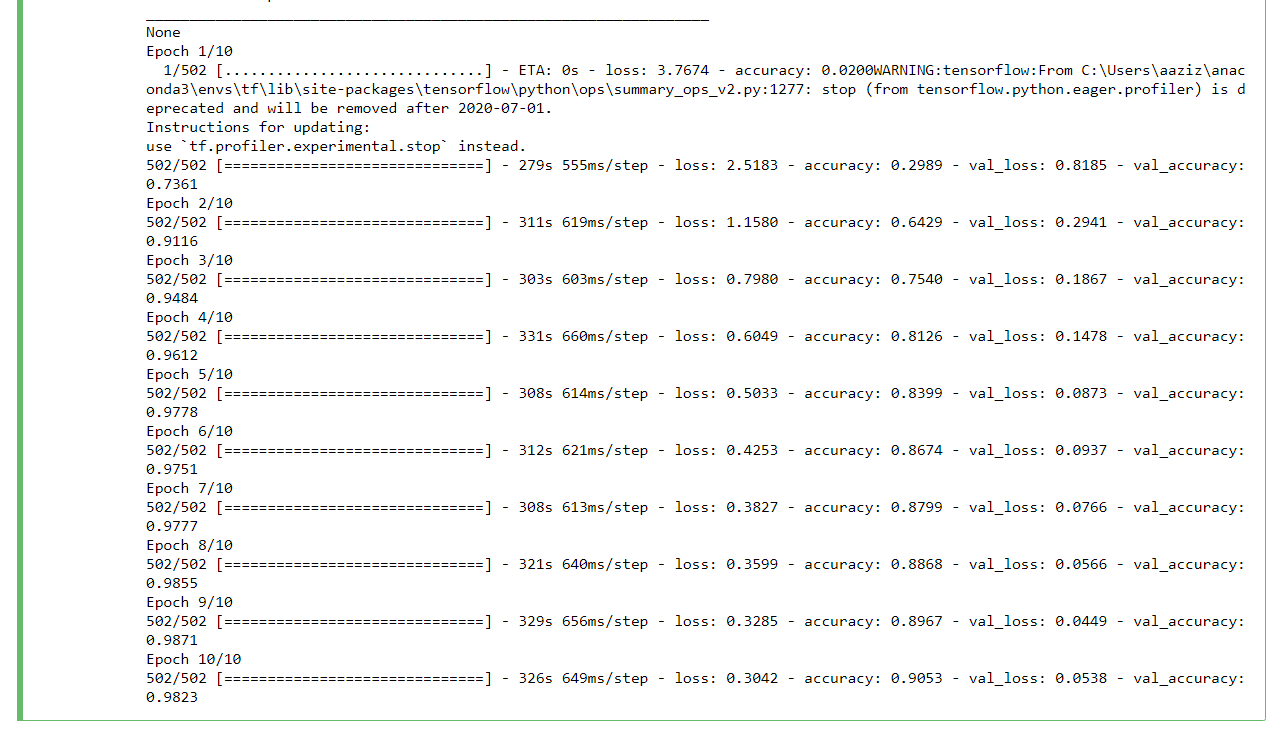


Figure5: training model with callbacks

In this figure we can see that the training time for each epochs is on average 310s and our total processing time is about 30 minutes.

We can see that accuracy is 0.91 which is fairly good and then we have the loss which is 0.30 which is again fairly good and then the accuracy validation which is 0.98 which is extremely good.

So, we can conclude that the use of callback function helped us to reduce about 30 minute in our training time.

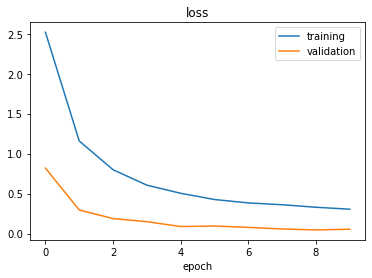
* We save our model to a file using this code:

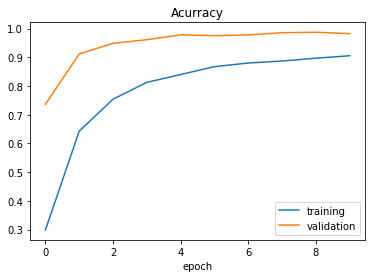
**model.save("my\_model")**

**model.save\_weights("weights.h5")**

* Loss and Accuracy plots on training and validation datasets for iterations (epochs).

These two images represent our lost an accuracy on training and validation dataset:

****

****

We can see that we are getting fairly good result, we can see that after about 6 ecpoch it is going at the same level so probably 6 to 8 epoch will be a good estimation of where we want to learn.

Problems Encountered

While building and training your deep neural network model, we was faced of some of the problems:

* Training was extremely slow. To train our dataset, it took us more than hour with only 10 iterations or epochs then we found out a solution of that by using a callback which helps us to reduce the training time more than half.
* We couldn’t save our model to a pickle file by using pickle, when we were trying it, we had this error: **cannot pickle '\_thread.RLock' object**. Then we used **model.save()** to do it.

**Performance Comparison**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Reference | Total sign | Loss (%) | Accuracy (%) | Processing time (s) |
| [16] | 123 | - | 95 | 0.43 |
| [17] | 340 | - | 90.27 | 0.35 |
| [18] | 2416 | - | 99.46 | - |
| Proposed method | 3500 | 27.1 | 91.43 | 0.40 |

Table2: comparison between proposed method and several existing methods

A comparison of previous studies in detecting the traffic sign is given in Table 2. It can be observed that CNNs used in [16] has the highest accuracy of over 99.42%, the processing time which is 0.43 is the highest one. The SVM use in [17] is with 340 signs has the lowest accuracy which is 90.27 and the lowest processing time 0.35 which is fairly good. In the propose method, we have the highest number of signs which is 3500, it has the third good result of accuracy and processing time which are 91.43 and 0.40. In the proposed method the result is good compared to other systems because of the highest number of total sign.

1. **Conclusion**

We covered how convolutional neural network can be used to classify traffic signs using convolutional neural network using Opencv in real-time using a simple webcam with high accuracy, and trying different model architectures. We built highly configurable code and developed a flexible way of evaluating multiple architectures. Our model reached close to 90% accuracy on the test set, achieving 98% on the validation set.

Personally, I thoroughly enjoyed this project and gained practical experience using Opencv, Keras, Tensorflow, matplotlib, and investigating artificial neural network architectures. Moreover, I delved into some papers in this field, which reinforced my understanding and more importantly refined my intuition about CNN.

In the future, I believe higher accuracy can be achieved by applying further regularization techniques such as batch normalization and also by adopting more modern architectures such as GoogLeNet’s Inception Module, ResNet, or Xception.

1. **References**

[1] P. Dewan, R. Vig, N. Shukla and B. K. Das, “An Overview of Traffic Signs Recognition Methods,” International Journal of Computer Applications, Vol. 168 – N..11, June 2017

[2] D. Jianmin and V. Malichenko, “Real time road edges detection and road signs recognition,” IEEE International Conference on Control, Automation and Information Sciences (ICCAIS), Changshu, China, 29- 31 Oct. 2015

[3] Y. Han, K. Virupakshappa, E. Vitor, S. Pinto and E. Oruklu, “Hardware/Software Co-Design of a Traffic Sign Recognition System Using Zynq FPGAs,”, In Electronics journal, 2015, Vol. 4, p. 1062- 1089; doi:10.3390/electronics4041062.

[4] F. Zaklouta and B. Stanciulescu, “Real-Time Traffic-Sign Recognition Using Tree Classifiers,” IEEE Transactions On Intelligent Transportation Systems, Vol. 13, N. 4, December 2012, p. 1507-1514.

[5] K. Tohidul Islam, R. Gopal Raj and G. Mujtaba, “Recognition of Traffic Sign Based on Bag-of-Words and Artificial Neural Network,” Symmetry journal, 2017, Vol. 9, 138; doi:10.3390/sym9080138.

[6] S. Maldonado-Bascón, S. Lafuente-Arroyo, P. Gil-Jiménez, H. GómezMoreno, and F. López-Ferreras, “Road-Sign Detection and Recognition Based on Support Vector Machines,” IEEE Transactions On Intelligent Transportation Systems, Vol. 8, N. 2, June 2007; p. 264-278.

[7] L. Abdi, “Deep learning traffic sign detection, recognition and augmentation,” Proceedings of the Symposium on Applied Computing, Maroc, 2017, p. 131-136.

[8] Y. Moualek, “Deep learning pour la classification des images,” Master’s thesis, Abou Bakr Belkaid University, Tlemcen, 2017.

[9]<http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset>. [10] Y. Le Cun, L. Bottou, Y. Bengio and P.Haffner, “Gradient-Based learning applied to document recognition,” Proceedings of IEEE, Vol. 86, N°11, p. 2278-2324, 1998.

[11] N. Srivastava, G. Hinton, A. Krizhevsky, I. Stuskever and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” Journal of Machine Learning Research, Vol. 15, 2014, p. 1929-1958.

[12] <https://www.tensorflow.org>.

[13] Python Project on Traffic Signs Recognition with 95% Accuracy using CNN & Keras  
<https://data-flair.training/blogs/python-project-traffic-signs-recognition/>

[14] Opencv documentation <https://opencv.org/>

[15] <https://www.guru99.com/tensorflow-vs-keras.html>

[16] ] [2] S. B. Wali, M. A. Hannan, A. Hussain, S. A. Samad, “An Automatic Traffic Sign Detection and Recognition System Based on Colour Segmentation, Shape Matching, and SVM” Mathematical Problems in Engineering, 2015, https://www.hindawi.com/journals/mpe/2015/250461/

[17] M. A. García-Garrido, M. Ocaña, D. F. Llorca, M. A. Sotelo, E. Arroyo, and A. Llamazares, “Robust traffic signs detection by means of vision and V2I communications,” in *Proceedings of the 14th IEEE International Intelligent Transportation Systems Conference (ITSC '11)*, pp. 1003–1008, IEEE, Washington, DC, USA, October 2011

[18] Stallkamp, M. Schlipsing, J. Salmen, C. Igel, “Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition”, Neural Networks, 2012, <https://www.sciencedirect.com/science/article/abs/pii/S0893608012000457>