**TRAFFIC SIGN RECOGNITION BY USING CONVOLUTIONAL NEURAL NETWORK**

**Kaan Berke UĞURLAR**

Istanbul Aydin University Software Engineering Department Istanbul,Turkey*, kaanberkeugurlar@gmail.com*

***Abstract***

Traffic sign recognition is a touchstone for various technologies (i.e., autonomous cars, etc.). In this paper, it will be shown the efficiency of using a convolutional neural network and which architecture is faster for optimal results.

***Keywords:*** Traffic Sign Recognition, Computer Vision, Deep Learning, Convolutional Neural Network.

**1. *Introduction***

Traffic signs—as known as road signs, are erected on the roads or on the edge of the roads. They provide information to all chauffeurs to inform them for preventing undesirable situations from happening. There are various traffic signs are being used all over the globe and it cannot be expected from a chauffeur to know every countries’ own traffic signs, but on the other hand, artificial intelligence is capable of recognizing any traffic sign which has been provided before—in such cases, there may be no need to provide beforehand. The difficulty of the problem depends on the number of data that has been supplied.

With traffic volumes increasing since the 1930s, many countries have adopted pictorial signs or otherwise simplified and standardized their signs to overcome language barriers and enhance traffic safety. Such pictorial signs use symbols (often silhouettes) in place of words and are usually based on international protocols. Such signs were first developed in Europe and have been adopted by most countries to varying degrees [[1]](#TrafficSign).

***Convolutional Neural Network***

Convolutional Neural Networks are very similar to ordinary Neural Networks. They are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g., SVM/Softmax) on the last (fully connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply [[2]](#ConvNetsRef2).

Convolutional neural networks, also called ConvNets, were first introduced in the 1980s by Yann LeCun, a postdoctoral computer science researcher. LeCun had built on the work done by Kunihiko Fukushima, a Japanese scientist who, a few years earlier, had invented the neocognitron, a very basic image recognition neural network.

The early version of CNNs, called LeNet (after LeCun), could recognize handwritten digits. CNNs found a niche market in banking and postal services and banking, where they read zip codes on envelopes and digits on checks [[3]](#OrifinofConvNetsRef3).

***2. Related Work***

Several experimental algorithms have been studied on the traffic signs recognition. Many developers have tried to solve this problem. They have used regular machine learning algorithms (i.e., SVM, Decision Trees, XGBoost and MLPhybrid) and various deep learning models to find the most efficient result in terms of time, performance and place the model takes. One of the best and smoothest examples of the relevant topic is Safat B. Wali’s SVM application in “An Automatic Traffic Sign Detection and Recognition System Based on Colour Segmentation, Shape Matching, and SVM” [[4]](#TrafficSignDetectionSVMRef4) paper to recognize traffic signs which was resulted by getting %95.71 accuracy, overall.

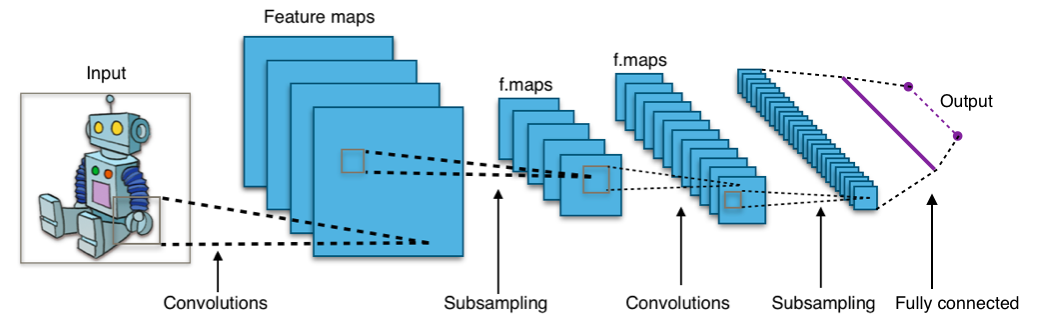


Figure 2.1: Convolutional Neural Network Representation [[5]](#ConvNetPicRef5)

Although the traffic sign recognition topic hasn’t been the hottest topic for a quiet while, there are many people who has been tried to improve the solving to this problem by using multilayer perceptron. Later on, people have decided to use convolutional neural networks are better fit. The authors of the “Traffic signs recognition with deep learning” paper [[6]](#TrafficSignRecognitionDLRef6), which was released in 2018, are claim that they have achieved %97.1 accuracy by applying dropout layers on the fully connected layers in less than 120 epochs with only 1000 images by applying image augmentation.

***3. Overview, Methods and Tools***

The convolutional neural networks start by getting images. Images must be the same size to train the model. Afterward, the number of feature maps must be generated by applying the kernel—also its size must be provided in advance— by starting most top-left cell of the image. Kernel roams over all the cells on image by spacing 1 if no stride hasn’t been told. If all feature maps are generated, then model make weights to head the next layer, in this case its batch normalization, which was introduced by Sergey Ioffe, Christian Szegedy in 2015 [[7]](#BatchNormalizationRef7), to make artificial neural networks faster and more stable through normalization of the input layer by re-centering and re-scaling [[8]](#BatchNormalizationRef8).

As it is a known fact, an activation function must be used in the layers. The architectures are introduced in this paper will always use activation functions after batch normalization layers. In this case, batch normalization layers’ outputs are input of activation function layers. The best results are achieved by using ReLu activation functions for all layers except the last fully connected layer that applies softmax activation function. Moreover, after all of the activation functions, a dropout layer with %50 percent hyperparameter has been used to wipe %50 of all the weights in order to prevent overfitting which has let the best result among all the hyperparameter testing.

Finally, last fully connected layer was used to determine the result—which traffic sign in this case. Softmax activation function makes it easier to calculate the probability of the predictions for understanding the errors better and visualizing them understandable.

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

***4.1 Architectures and Models***

There were several architectures trained to choose the best result in terms of time, performance and place the model takes, but best of two architectures are being exposed here to be inspected and to be compared.

In the Convolutional Neural Network Architecture No-10 ([Fig. 4.1](#Fig4dot1)), there are 16,322,475 total parameters but, 1,792 are not trainable which is the number of parameters of Batch Normalization layers’ in half.

Calendar

Description automatically generated with medium confidence

Figure 4.1: Convolutional Neural Network Architecture No-10

In the Convolutional Neural Network Architecture No-11 ([Fig. 4.2](#Fig4dot2)), there are 5,771, 179 total parameters, but 1,792 are not trainable which is the number of parameters of Batch Normalization layers’ in half. It is obvious that the architecture number 11 is using almost 1/3 of the parameters of the architecture number 10. There must be a trade-off and this comparison will be held in the Section 5.

Calendar

Description automatically generated with medium confidence

Figure 4.2: Convolutional Neural Network Architecture No-11

***4.2 Data***

Data has been used to train these all models are provided by GTSRB section of INI Benchmark Website [[9]](#GermanTrafficSignDataRef9). The data belongs to a single-image, multi-class classification problem, which has been provided more than 50,000 images and 43 different classes. Images have different size attributes (width/height) but as it was mentioned before on Section 3, all images must be the same size. All images were preprocessed before loaded into the model and they were resized as 30x30.

***4.3 Used Tools***

For the project, several tools were used to achieve optimal result. They were listed below.

|  |  |
| --- | --- |
| **Tool Name** | **Version** |
| Python | 3.8 |
| Pycharm | 2020.2.2 |
| Pip | 20.2.4 |
| Pyenv | 1.2.21 |
| Tensorflow | 2.3.1 |
| Numpy | 1.18.5 |
| OpenCV | 4.4.0.46 |
| Sklearn | 0.24.0 |
| Matplotlib | 3.3.3 |

Table 4.1: Used Tools and Their Versions

***4.3 Used Hardware***

It takes too many hours to developing a project like “Traffic Sign Recognition” with a classical CPU using but thanks to new improvements in science, a model can be trained in minutes instead of days by taking advantages of GPUs.

In this project Google’s Colab product was used and thanks to Google, Colab lets people use $70,000-dollar worth GPU to develop any project they want, and they do not charge a penny for it. To creating architectures, a MacBook Pro (15-inch, 2019) with 2,6 GHz 6-Core Intel Core i7 and 16 GB 2400 MHz DDR4 has been used and the vast majority of the models trained first on that machine. If any problem hasn’t been occurred, then the code moved to Colab Notebook to train model, end-to-end.

**5. *Analysis, Experiments and Design***

There are 13 models were trained to select the best one in terms of time, performance and place the model takes. Finally, the model no-10 was selected as the best one. Between model no-10 and model no-11 there are huge difference between number of parameters. Then, the question is why model no-11 haven’t been selected over model no-10. As it was mentioned in Section 4.1, there is a tradeoff, which it can be inspected in Table 5.1, and that tradeoff would have mattered if the time or place the model takes would be more valuable than accuracy.

In this project, traffic sign recognition is not developed for an autonomous car. Then, accuracy is much more valuable. As can be seen, model no-10 has 0.0193 validation loss whilst model no-11 has 0.0196 and there is exactly 8 minutes difference in training time.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model No** | **Total Number of Parameters** | **Seconds per Epoch** | **Best val\_loss** | **Best val\_loss's Epoch Number** |
| 1 | 39.260.587 | 18 | 0.0915 | 7 |
| 2 | 10.932.139 | 16 | 0.0858 | 42 |
| 3 | 6.428.955 | 7 | 0.0530 | 9 |
| 4 | 5.767.595 | 3 | 0.0304 | 39 |
| 5 | 5.767.595 | 4 | 0.0232 | 18 |
| 6 | 16.318.891 | 6 | 0.0199 | 65 |
| 7 | 4.784.555 | 3 | 0.0393 | 45 |
| 8 | 1.638.827 | 2 | 0.0669 | 100 |
| 9 | 16.318.891 | 6 | 0.0329 | 47 |
| 10 | 16.322.475 | 8 | 0.0193 | 134 |
| 11 | 5.771.179 | 4 | 0.0196 | 148 |
| 12 | 8.994.859 | 25 | 0.0242 | 140 |
| 13 | 2.580.523 | 9 | 0.0236 | 164 |

Table 5.1: Model Inspections and Comparisons

Following part involves only two figures ([Figure 5.1](#Fig5dot1), [Figure 5.2](#Fig5dot2)). Even though those two figures are self-explanatory, there might be some things to reveal. Loss is the penalty for a bad prediction, and it has to go down over the iterations. On the other hand, accuracy is the percentage of instances that are correctly classified, and it has to go up over the iterations. As can be seen, both loss and accuracy were improved over epochs, smoothly.

Chart

Description automatically generated

Figure 5.1: Convolutional Neural Network Architecture No-10 loss/val\_loss Plot

Chart

Description automatically generated

Figure 5.2: Convolutional Neural Network Architecture No-10 acc/val\_acc Plot

**6. *Conclusion***

In this article, a various number of convolutional neural network architectures were used to solve traffic signs recognition problem by using Python language and compared the results ([Table 5.1](#ModelComparison)). According to the results, the best accuracy to identify traffic signs were achieved by applying Convolutional Neural Network Architecture No-10. The project can be taken further by applying transfer learning and data augmentation techniques, but the result is satisfying for the project and training time should not take too much as Convolutional Neural Network Architecture No-10 takes less than 18 minutes by using the GPU provided by Google’s Colab product.

***REFERENCES***

[1] Traffic sign, <https://en.wikipedia.org/wiki/Traffic_sign>

[2] Convolutional Neural Networks, <https://cs231n.github.io/convolutional-networks/>

[3] Origin of ConvNets, <https://bdtechtalks.com/2020/01/06/convolutional-neural-networks-cnn-convnets/>

[4] An Automatic Traffic Sign Detection and Recognition System Based on Colour Segmentation, Shape Matching, and SVM, <https://www.hindawi.com/journals/mpe/2015/250461/>

[5] Convolutional Neural Network Representation, <https://towardsdatascience.com/nlp-with-cnns-a6aa743bdc1e>

[6] Traffic signs recognition with deep learning, <https://www.researchgate.net/publication/331847325_Traffic_signs_recognition_with_deep_learning>

[7] Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, <https://arxiv.org/abs/1502.03167>

[8] Batch Normalization, <https://en.wikipedia.org/wiki/Batch_normalization>

[9] The German Traffic Sign Recognition Benchmark, <https://benchmark.ini.rub.de/gtsrb_news.html>