Traffic Sign Recognition using CNN

Youhana Mikhaiel
Department of Communications and Information Engineering
University of Science and Technology at Zewail City
Giza, Egypt
s-youhana.mikhaiel@zewailcity.edu.eg

Dr. Elsayed Hemayed
Department of Communications and Information Engineering
University of Science and Technology at Zewail City
Giza, Egypt
ehemayed@zewailcity.edu.eg

***Abstract*—Machine learning algorithms are widely used nowadays in many life applications. One of these applications is traffic sign recognition. There are various traffic signs different in their visual appearance. These signs could be easily identified by humans unless certain conditions occurred such as driving fast, different orientation or weather conditions so a need for an automated system to detect and recognize these signs must be available in order to work for autonomous cars system. Deep learning and especially convolutional neural networks (CNN) has proved to be more efficient giving higher accuracies.**

Keywords—CNN, object recognition, deep learning.

I. INTRODUCTION

Deep learning and especially CNN was proved to be one of the most successful techniques in the field of automation and artificial intelligence for visual systems. We will concentrate on one of the research areas in this field, which is recognition of traffic signs in autonomous cars in order to take certain actions according to these signs such as keeping the lane, speed limit and dangerous slope. Traffic signs follow some design principles so these signs could be categorized according to their shape, color, icon and text. The problem here lies in the different appearances these traffic signs seems to have due to rotations, partial occlusion, different illuminations and weather conditions. In this paper, we are going to try CNN for our experiment and we will try it using different parameters such as the type of the activation function and the number of convolutional layers. We will compare all these results together in order to achieve the highest accuracy, as this application is a very critical real-time application, which needs accuracy reaching approximately 100%.

II. DATASET

A. Description

Our dataset represents images for different traffic signs generated by German Traffic Sign Recognition Benchmark (GTSRB) where there are 51840 different images of 43 distinct traffic signs with different frequencies. This dataset was created from 10 hours of recording a video in a car driven in different road types in Germany recorded in three different months; March, October and November so we can see that these photos were taken in various circumstances such as illumination changes, different orientations, weather conditions and partial occlusions as shown in figure 1.

These variations makes our dataset more challenging. Every number of those signs could be categorized according to either their shape such as triangle and circle or color. This technique of categorization helps humans in detecting these signs and identifying it with very high accuracy but in our project we are aiming to do this using convolutional neural networks (CNN).

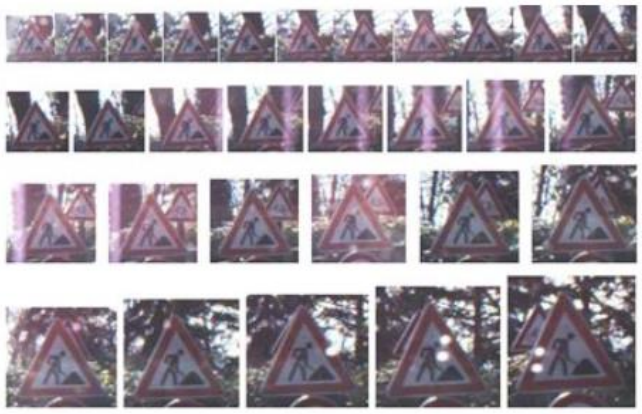


Figure 1: Traffic signs in different conditions

From figure 2, we can see that there are two main shapes for the signs, which are the triangle and the circle with two signs only having different shapes; one is a hexagon and the other one is a diamond. After that, we can make subcategories according to the sign's most dominant color or the color combinations to even divide the dataset more and increase the number of features correlated to certain classes and not correlated to others, which would contribute in decreasing the error in identifying this sign. The 51840 instances were divided into 75% as training set and 25% as test set so the training set consists of 39210 instances while the test set consists of 12630 instance. Both training and test datasets are labeled.

Figure 2: Traffic signs 16 categories

B. Preprocessing

In order to just prove the concept and to decrease the processing time, all images were resized to be 32x32 pixels instead of an average of 100x100 pixels and are converted from RGB to grayscale images. The input samples before and after the preprocessing are shown in figures 3 and 4.

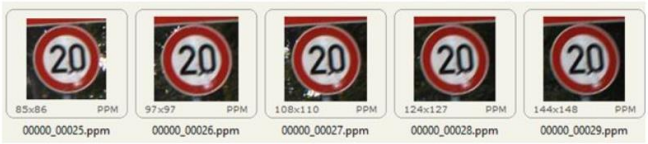


Figure 3: Input samples before preprocessing

Figure 4: Input samples after preprocessing

III. CONVOLUTIONAL NEURAL NETWORKS (CNN)

Multi-scale CNN is an extension of MLP which is inspired by the visual cortex. Regarding the traditional feedforward neural network in image recognition, it deals with pixels that are close to each other like those far away from each other, so it suffers from the curse of dimensionality due to the full connectivity between the network nodes and they can't scale well to higher resolution images.

So, instead of using the complete forward neural network connection with different weight for each connection, CNN concept saved a lot of this computational work. It depends on the correlation between features in the input data like audio, video or image. Thus, the same weights can be scanned (convolved) throughout the whole input data in order, which saves a lot of time and get great results. It can be found that CNN is not useful for the input data whose features' order is not important like detecting breast cancer which depends on age, gender, habitat, etc.. The order of these features is not important so CNN is not recommended in these cases.

A. Convolutional layer

A filter is used with a smaller size than that of input data (image). This filter is considered as the weights of the neural network which are updated in training stage. It's convolved through the whole input image by a dot product to get an output of the same size as the input (width and height) but a third dimension for the output is updated according to the number of filters used.

B. Activation layer

An activation function is applied to the neurons output. It increases the nonlinear properties of the whole network without affecting the convolutional layer. There are many types of activation functions like ReLu function (Rectified Linear Units) which can be expressed as f(x) = max(0, x) and sigmoid function which can be expressed by

$$f(x) = (1+ e^{(-x)})^{-1}$$

C. Max-pooling layer

This layer partitions the input image into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum. The intuition is that the exact location of a feature is less important than its rough location relative to other features. The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters and amount of computation in the network, and hence to also control over fitting.

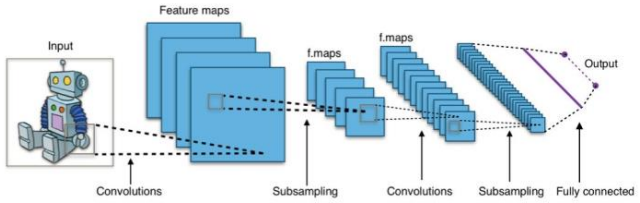
D. Fully-Connected layer

Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

E. Dropout

Fully-Connected layer is prone to over fitting. At each training stage, individual nodes are either "dropped out" of the net with probability 1-p or kept with probability p, so that a reduced network is left. Only the reduced network is trained on the data in that stage. The removed nodes are then reinserted into the network with their original weights.

A summary diagram of CNN is shown in figure 5.



IV. EXPERIMENT

Our benchmark experiment was done by training a CNN consisting of two convolution layers followed a max-pooling layer and then a fully connected layer. All the outputs of these layers pass by a ReLu activation function. Number of epochs is 6 and the dropout value is 0.25.

A. Experiment 1

In this experiment, we managed to change the dropout value from 0.25 to 0.5 in order to avoid any over fitting but we found no big effect on accuracy and it took more time.

B. Experiment 2

In this experiment, the activation function was changed from ReLu to sigmoid with the same number of convolutional layers and epochs. Dropout equal to 0.25. The accuracy seems to decrease with more time taken.

C. Experiment 3

In this experiment, we increased the number of convolution layers to be three with the 6 epochs and dropout equal to 0.25 and ReLu activation function. The accuracy increased a little bit.

D. Experiment 4

In this experiment, the number of convolutional layers are still three layers with dropout equal to 0.25 and using ReLu activation function but the number of epochs was increased from six to fifteen. We could notice better accuracy with more time taken.

E. Experiments 5,6

In these experiments, the number of epochs remained fifteen with ReLu and dropout equal to 0.25 but we

increased the number of convolutional layers to be four in experiment 5 and five convolutional layers in experiment 6. We could notice a better accuracy reaching more than 99% of cross-validation accuracy with total time of training about 100 minutes.

V. RESULTS

Below is the table of the results of all experiments done with both cross-validation accuracy and test accuracy.

Exp#	bench-mark	1	2	3	4	5	6
Dropout	0.25	0.5	0.25	0.25	0.25	0.25	0.25
# conv layers	2	2	2	3	3	4	5
Activation function	ReLu	Sigmoid	ReLu	ReLu	ReLu	ReLu	ReLu
Validation accuracy	96.8	96	63.6	97.8	98.8	98.9	**99.1**
Test accuracy	91	91.2	57.4	94.4	95.6	96.1	**96.5**
# epochs	6	6	6	6	15	15	15
av. epoch time in mins	2.5	3	5	4	4.5	5	6

VI. DISCUSSION

From the results shown in the above table, we can see that with increasing the number of convolution layers, the accuracy increases but not for long. When we tried to increase the number of convolutional layers to six, the accuracy began to decrease. When changing the activation function from ReLu to sigmoid, the accuracy also witnessed a huge drop and the time increased so much. This may have happened due to sparsity and reduced likelihood of vanishing gradient. Increasing number of epochs leads to more training and more accuracy but more total time.

VII. CONCLUSION

We can conclude that CNN are very powerful method in visual applications where is can be used to compensate for the human eye in autonomous cars leading to high driving efficiency due to high accuracies achieved from these trained neural network models.

VIII. FURTHER DEVELOPMENT

In this experiment, we just wanted to do a proof of concept but for a real-time traffic sign recognizer, we need an algorithm to detect the traffic sign and extract it from the scene and then test it using our model. Our dataset signs have known shapes such as circles, triangles, diamonds and hexagons. We can detect such shapes using a very famous algorithm called Hough transform.

REFERENCES

[1] *Stallkamp J., Schlipscing M., Salmen J., Igel C., (2012) "Man vs. Computer: Benchmarking machine learning algorithms for traffic sign recognition" Retreived from Elsevier, Volume 32 Pages 323-332*