Traffic Sign Classifier

Vignesh Sivanandha Rao

*Missouri State University*

*901 S National Ave*

*Springfield, MO 65897*

Abstract

In the present world, the need for road safety has been increasing drastically due to the increase in the rate of accidents as well as vehicle production. Many modern vehicles have the facility of recognizing some of the traffic signs by the road, which can assist the drivers to improve road safety. This project uses neural network layers to train and classify traffic sign images using convolutional neural network.

*Keywords:* Convolutional Neural Network, Lenet, Batch Normalization, Data Augmentation, Dropout, Adam, Sophistic Gradient Descent

1. Introduction

In this project, I used neural network layers to train and classify traffic sign images. I used LeNet architecture to train an existing model with different datasets and to compare the results. The dataset was gathered using images that were manually clicked on the roadside combining with the images randomly downloaded from google using image downloader plugin for google chrome and resized them to reduce the size of the dataset. The following libraries were used in this project, OpenCV for data augmentation, tensorflow for building neural network layers, pickle to create/merge datasets in the form of matlab files. Pandas to read csv files containing the class IDs.

* 1. Methods and Functions

This section of the paper will talk more about the methods/functions, parameters and techniques I used in implementing this project.

Regularization:

* Batch Normalization
* Data Augmentation
* Dropout

Optimization:

* Adam
* SGD

**Batch Normalization:**

Normalization is the process of converting unstructured images to structured images. In my case, I had images of different sizes and pixels. So I used Image Normalization to convert all the images to a unified dimension and standard pixel rate. This was achieved by dividing the largest value possible by the image type, not the actual image itself which was further scaled by 255.

● normalized = (image.astype(np.float32) - 128) / 128

**Data Augmentation:**

Data Augmentation is the process of replicating or creating new data from the original data, in my case I recreated images from the original image. It can be applied to any form of data but may be especially useful where additional information can help provide more in-depth insight. In this project I transformed the original image into 3 other images by scaling, translation and rotation.

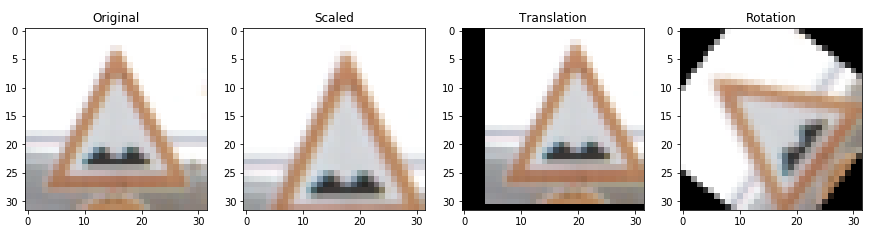


Fig 1.1: Results of Data Augmentation

The plot below shows us that I recreated 80,451 new images from the original 26,817 images after data augmentation:

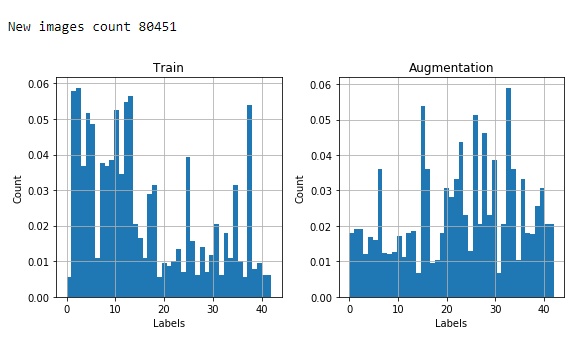


Fig 1.2: Label distribution of data augmentation and new image count post augmentation

* 1. Experiment:

I used a dataset which was a mix of photos shot manually as well as existing dataset trained the model with learning rate as 5, batch size as 128 and epoch values as 20. The results were not satisfactory because of high learning rate. After learning more about optimizers, I lowered the learning rate because of which I saw better results in training and validation accuracy. Best results were achieved with learning rate as low as 0.0025 between 23 and 25 epochs. I also used different datasets for training to see the factors of a dataset that affects the accuracy. Using data visualization on these datasets, I found that less clear images, images taken from a different angle had considerably low accuracy.

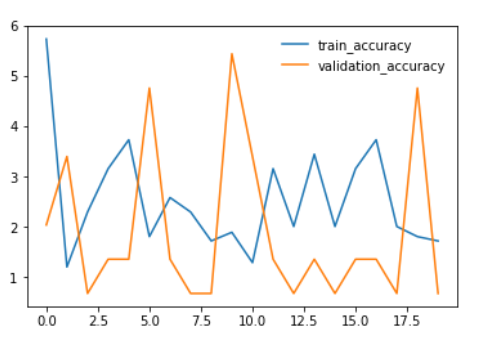
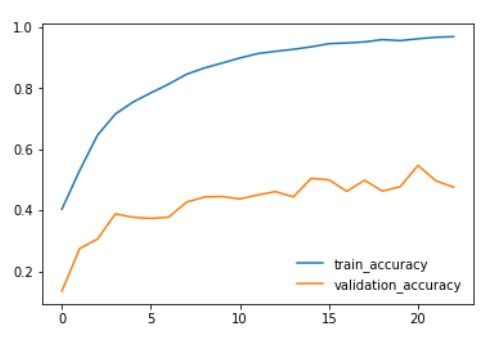
 

Fig 1.3: Initial results Fig 1.4: Better results with low learning rate

* 1. Classification of Adam and SGD Optimizer:

Both Adam and SGD are optimization algorithms. Adam is an improved version or update of SGD optimizer for less epoch values. Adam computes individual learning rates from different parameters whereas SGD maintains single learning through different parameters. SGD is also known as an incremental gradient as most SGD techniques run in iterative fashion.

I used both the optimizers in this project and did a classification to see the factors that affect performance in terms of optimizers. Initially I trained the model using adam optimizer with low learning rate which gave us results as good as 99.62 in training accuracy and 80.23 in validation accuracy. With the same parameters I saw pity results as 15.84 in training accuracy and 5.33 in validation accuracy while using SGD optimizer. To learn more, I increased the epoch to 251 using SGD optimizer where the validation accuracy reached between 32 and 34 and training accuracy as 82.

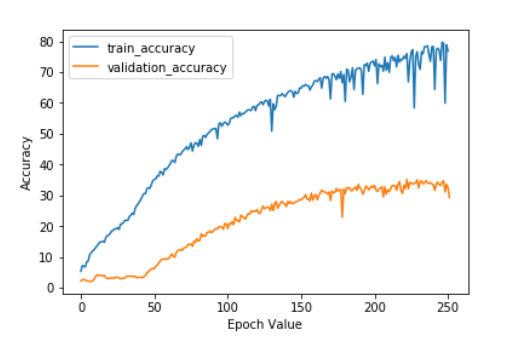


Fig 1.5: Result of Stochastic Gradient Descent (SGD)

Conclusion

From this project I gained knowledge on the working of deep neural networks and learnt how optimizers affect the accuracy of a model. I also achieved a basic classification of SGD and Adam optimizer.

Acknowledgements

I would like to thank Dr. Anthony Clark for teaching the concepts of deep neural network and convolutional neural network and for the excellent guidance and encouragement. I would also like to thank Jeremy Shannon for the initial code on github implementing Lenet architecture on traffic sign classifier.

References

1. Alexander Shustanova, Pavel Yakimova, CNN Design for Real-Time Traffic Sign Recognition, 3rd International Conference “Information Technology and Nanotechnology”, ITNT-2017, 25-27. doi: 10.1016/j.proeng.2017.09.594.
2. Canyong Wang, Research and Application of Traffic Sign Detection and Recognition Based on Deep Learning, 2018 International Conference on Robots & Intelligent System (ICRIS), doi: 10.1109/ICRIS.2018.00047
3. https://github.com/jeremy-shannon/CarND-Traffic-Sign-Classifier-Project/blob/master/Traffic\_Sign\_Classifier.ipynb