Traffic Sign Classification with Deep Learning

Andrii Bezmen
Deep Learning Class
Computer Science Department
Hood College
ab44@hood.edu

1. Introduction

Traffic signs classification is one of the foremost important integral parts of autonomous vehicles and advanced driver assistance systems (ADAS). Most of the time driver missed traffic signs due to different obstacles and lack of attentiveness. Automating the process of classification of the traffic signs would help reducing accidents. Traditional computer vision and machine learning based methods were widely used for traffic signs classification, but those methods were soon replaced by deep learning based classifiers. Recently deep convolutional networks have surpassed traditional learning methods in traffic signs classification. With the rapid advances of deep learning algorithm structures and feasibility of its high performance implementation with graphical processing units (GPU), it is advantageous to re-look the traffic signs classification problems from the efficient deep learning perspective. Classification of traffic signs is not so simple task, images are effected to adverse variation due to illumination, orientation, the speed variation of vehicles etc. Normally wide angle camera is mounted on the top of a vehicle to capture traffic signs and other related visual features for ADAS. This images are distorted due to several external factors including vehicles speed, sunlight, rain etc. Sample images from GTSRB dataset are shown in Fig. 1. German Traffic Sign Recognition Benchmark (GTSRB) is one of the reliable datasets for testing and validating traffic sign classification and detection algorithms. In the competition of GTSRB, top-performing algorithm exceeds best human classification accuracy. By using committee of neural networks Ciresan et al [15] achieved highest ever performance of 99.46% , which surpassed the best human performance of 98.84%.

Traffic sign classification becomes a mature area with the increasing focus on autonomous driving research. Notable research work exists on detection and classification traffic signs for advanced driver assistance systems. Most of the works attempted to address the challenged involved in real life problems due to scaling, rotation, blurring etc. The purpose of this project is to classify traffic signs present in the image into one of the 43 different categories. With this model, I will be able to read and understand traffic signs in real time which are a very important task for all autonomous vehicles.

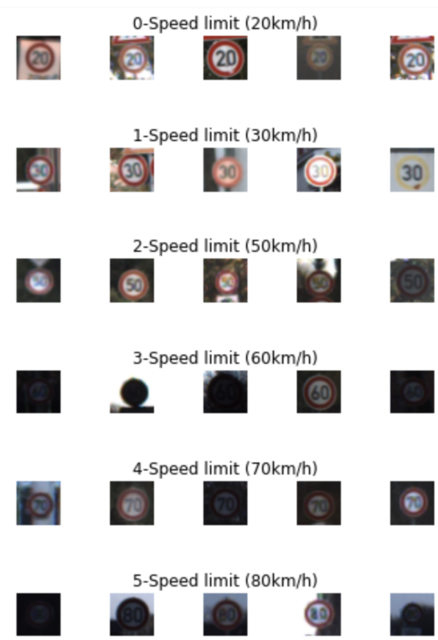

Figure 1. Sample images from GTSRB dataset

2. Data and Model

2.1. Data

Traffic sign classification is the process of automatically recognizing traffic signs along the road, including speed limit signs, yield signs, merge signs, etc. Being able to automatically recognize traffic signs enables us to build “smarter cars”. Self-driving cars need traffic sign recognition in order to properly parse and understand the roadway. Similarly, “driver alert” systems inside cars need to understand the roadway around them to help aid and protect drivers. The dataset I will be using to train my own custom traffic sign classifier is the German Traffic Sign Recognition Benchmark (GTSRB). The GTSRB dataset consists of 43 traffic sign classes and nearly 50,000 images. The traffic signs have been pre-cropped for me, implying that the dataset annotators/creators have manually labeled the signs in the images and extracted the traffic sign Region of Interest (ROI) for me, thereby simplifying the project.

There are a number of challenges in the GTSRB dataset, the first being that images are low resolution, and worse, have poor contrast (as seen in Figure 1 above). These images are pixelated, and in some cases, it's extremely challenging, if not impossible, for the human eye and brain to recognize the sign. The second challenge with the dataset is handling class skew as we can see on the Figure 2.

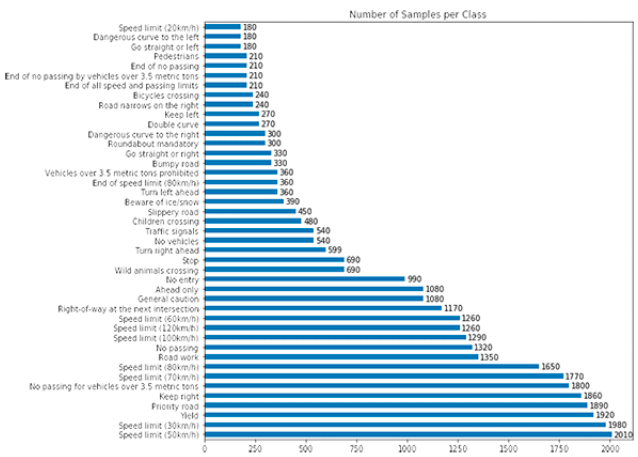

Figure 2. Class skew

The top class (Speed limit 50km/h) has over 2,000 examples while the least represented class (Speed limit 20km/h) has under 200 examples — that's an order of magnitude difference! In order to successfully train an accurate traffic sign classifier we'll need to devise an experiment that can preprocess our input images to improve contrast and account for class label skew.

2.2. Model

A CNN consists of a series of Convolutional and Pooling layers in the Neural Network which map with the input to extract features. A Convolution layer will have many filters that are mainly used to detect the low-level features such as edges of a face. The Pooling layer does dimensionality reduction to decrease computation. Moreover, it also extracts the dominant features by ignoring the side pixels. Preprocessing images before feeding into the model gives very accurate results as it helps in extracting the complex features of the image. OpenCV has some built-in functions like cvtColor() and equalizeHist() for this task. First, the images are converted to grayscale images for reducing computation using the cvtColor() function. The equalizeHist() function increases the contrasts of the image by equalizing the intensities of the pixels by normalizing them with their nearby pixels. At the end, I normalized the pixel values between 0 and 1 by dividing them by 255. After reshaping the arrays, it's time to feed them into the model for training. But to increase the accuracy of my CNN model, we will involve one more step of generating augmented images using the ImageDataGenerator. This is done to reduce overfitting the training data as getting more varied data will result in a better model. I'm also converting the labels to categorical values.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 60)        1560
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 24, 24, 60)        90060
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 60)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 30)        16230
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 30)          8130
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 30)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4, 4, 30)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 480)               0
_________________________________________________________________
dense_1 (Dense)              (None, 500)               240500
_________________________________________________________________
dropout_2 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_2 (Dense)              (None, 43)                21543
=================================================================
Total params: 378,023
Trainable params: 378,023
Non-trainable params: 0
```

Figure 3. Model

The model contains two Conv2D layers followed by one MaxPooling2D layer. This is done two times for the effective extraction of features, which is followed by the Dense layers. A dropout layer of 0.5 is added to avoid overfitting the data. After successfully compiling the model, and fitting in on the train and validation data, I evaluated it by using Matplotlib

Test Score: 0.02595394855437624
Test Accuracy: 0.9933907985687256

This shows that I fitted the data well keeping both the training and validation loss at a minimum.

3. Conclusion

In this work, I covered how deep learning can be used to classify traffic signs with high accuracy, employing a variety of preprocessing and regularization techniques (e.g. dropout), and trying different model architectures. I've built highly configurable code and developed a flexible way of

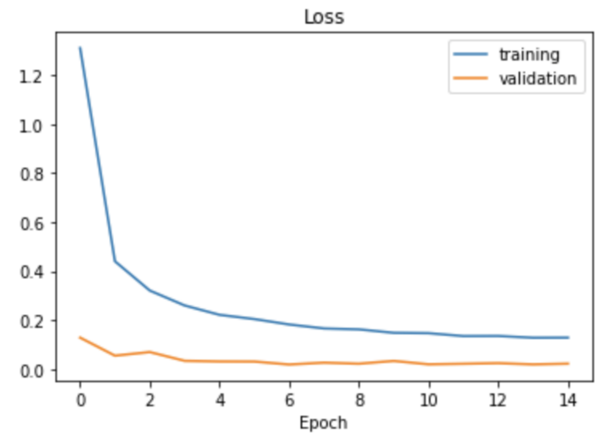

Figure 4. The Loss function

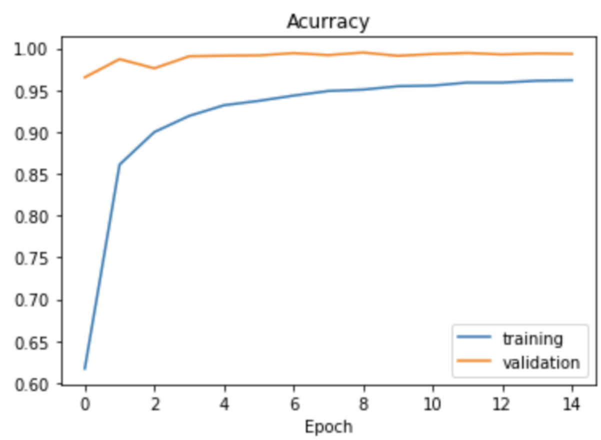

Figure 5. The Accuracy function.

evaluating multiple architectures. My model reached close to 98% accuracy on the test set, achieving 99% on the validation set.

4. References

[1] P. Sermanet and Y. LeCun, ”Traffic sign recognition with multi-scale Convolutional Networks,” The 2011 International Joint Conference on Neural Networks, 2011, pp. 2809-2813, doi: 10.1109/IJCNN.2011.6033589.

[2] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In Proceedings of the IEEE International Joint Conference on Neural Networks, pages 14531460. 2011.

[3] Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K. ”Spatial Transformer Networks”. arXiv preprint arXiv:1506.02025 (2015)

[4] Haloi, M. A novel pLSA based Traffic Signs Classification System. arXiv preprint arXiv:1503.06643 (2015).