CSC396 Final Project: Traffic Signs Prediction Model

Thy Nguyen, Thanh Nguyen, Steven Bogaerts

Department of Computer Science, DePauw University
Greencastle, IN 46135, U.S.A.
minhthynguyen_2024@depauw.edu
thanhnguyen_2025@depauw.edu
stevenbogaerts@depauw.edu

Abstract

In this paper, we describe multiple neural networks in the German Traffic Sign Recognition Benchmark. We use a simple neural network with a stack of convolution, max pooling, and dropout layers that achieves 95% accuracy on traffic signs classification.

1 Introduction

Electric vehicles are now becoming popular and will be the mainstream mode of transportation. Researchers have been working on self-driving cars recently to enhance the abilities of electric cars. As a result, designing a system that can recognize the traffic sign is essential. The visual appearance of traffic signs in real-world environments varies greatly. Changes in illumination, weather conditions, and partial occlusions affect how people perceive road signs [4]. Therefore, a large number of different traffic sign classes must be recognized accurately. Traffic signs are designed to be easily readable by humans, who excel at this task.

The development of deep learning and artificial intelligence can help us build a strong system with high accuracy in recognizing traffic signs. Researchers have built a lot of neural networks that perform image classification, which can also achieve high accuracy in classifying traffic sign recognition. In this paper, we mimic the architecture of VGG to classify traffic signs. VGG stands for Visual Geometry Group; it is a standard deep Convolutional Neural Network (CNN) architecture with multiple layers. Simonyan et al. proposed VGG-16 and VGG-19, very deep neural networks on large-scale images, which achieved the highest accuracy on ImageNet in 2015 [2].

As a result, we choose VGG to perform in the German Traffic Sign Recognition Benchmark dataset. Alvaro et. al. proposed a deep neural network that comprises convolutional layers and spatial transformer networks for traffic sign classification [1]. Their model achieved the highest accuracy on the German Traffic Sign Recognition Benchmark dataset at 99.71% accuracy on the test set. Despite not having higher accuracy on our test set, we experiment with many different neural networks to determine the best parameters.

2 Data

The German Traffic Sign Recognition Benchmark dataset contains 43 classes representing 43 traffic signs that appeared in German traffic [3]. The dataset contains 42,471 training images for 43 traffic signs and 12,630 for testing. The dataset is imbalanced for each class. There are 2,995 images for the "Keep Right" sign and 2,412 images for the "Speed Limit (50km/h)" sign, while there are only 211 images for the "Go straight or left" sign. The image of each traffic sign is resized to be 32×32 pixels. Figure 1 shows six images from a training set of six traffic signs. We can see that the data set provides images of various brightness, day and night, reflecting driving obstacles in real life. We read the training images using ImageData-

Generator from Tensorflow, which helps us scale the images, translate the images, or blur the images.

Figure 1: *Images from training set*

3 Experimental Setup

We set up multiple neural networks to examine whether more convolutional layers or more fully connected layers increase accuracy. We build a simple neural network containing four convolutional layers with a number of filters: 32, 64, 64, and 128. Each convolutional layer is followed by a max pooling layer and a dropout layer with a specific dropout proportion. We use the relu activation for each convolutional layer and the pool size of 2×2 for each max pooling layer to reduce the number of parameters by half. Afterward, we add two dense layers with the number of neurons 128 and 43 at the end of our model.

Firstly, we want to see which dropout proportions and optimizers perform well on the dataset. We construct five models using the RMSprop optimizer, with dropout proportions of 0, 0.1, and 0.2 etc., and five models using the Adam optimizer, with the same dropout proportions. These ten models utilize the same architecture as above.

Secondly, we want to examine adding more fully connected layers to our model. To evaluate them, we construct three models using the same number of convolutional layers and a dropout proportion of 0.2. Each model will have a different number of fully connected layers and a different number of neurons in those fully connected layers.

Finally, we examine how many convolutional layers give the highest accuracy on the test set. We construct four models that contain a different number of convolutional layers with the same dropout and Adam optimizer.

4 Experimental Results

4.1 Dropout Proportion and Optimizer

To determine the best dropout proportion and optimizer, we constructed the model as in Section 3. As a result, the increase in dropout proportion leads to an increase in accuracy on the validation and test sets for both optimizers. As we can see, the Adam optimizer achieves higher accuracy than RMSprop. The best neural network gets 95% accuracy on the test set that uses the Adam optimizer with a dropout proportion of 0.2.

Dropout Proportion	**Optimizer**	**Validation Accuracy**	**Test Accuracy**
0.0	RMSprop	0.896	0.927
0.0	Adam	0.903	0.933
0.1	RMSprop	0.912	0.933
0.1	Adam	0.916	0.949
0.2	RMSprop	0.922	0.931
0.2	Adam	0.916	0.950
0.3	RMSprop	0.885	0.894
0.3	Adam	0.907	0.931
0.4	RMSprop	0.815	0.846
0.4	Adam	0.908	0.933

Table 1: *Dropout Proportion and Optimizer Experiments*

4.2 Fully Connected Layers

We still construct the neural network as above but replace the number of fully connected or dense layers. In this experiment, we use the Adam optimizer with a dropout proportion of 0.2, because this model achieves the highest accuracy in the test set. The name of the model contains the letter "C" to denote convolutional layers, followed by the number of filters for each layer. The letter "d" stands for dropout proportion, and the letter "D" stands for dense layers, followed by the number of neurons per layer. As we can see in Figure 2, adding a 512-neuron dense layer makes the model perform worst on the validation set. On the other hand, using only 128-neuron and 43-neuron dense layers is not as good as adding 256-neuron dense layers. As a result, we should tune our hyperparameters for the number of dense layers carefully to avoid overfitting or underfitting and get the best accuracy.

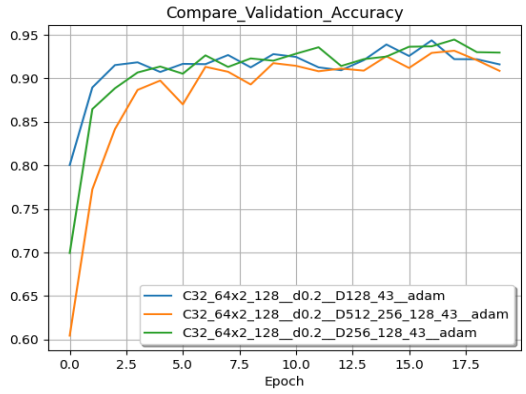

Figure 2: *Dense Layers Experiment*

4.3 Convolutional Layers

In this experiment, we use the Adam optimizer and a dropout proportion of 0.2. The dense layers contain two first layers with 256 and 128 neurons, and the last dense layer has 43 neurons with softmax activation.

In figure 3, we can see that adding more convolutional layers with more filters increases the accuracy of the test set. The model, which has four convolutional layers with a number of filters, 32, 64, 128, and 256, respectively, gives 95.01% accuracy on the test set.

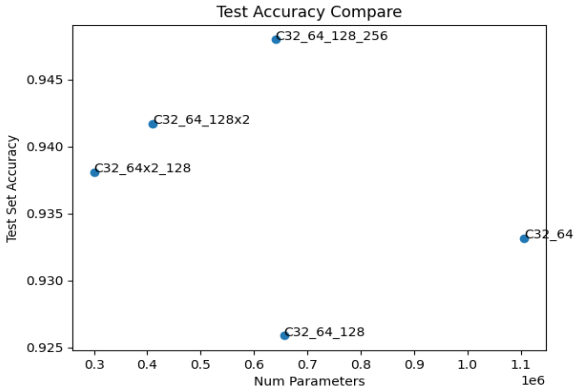

Figure 3: *Convolutional Layers Experiment*

5 Future Work

The data set is imbalanced for every class, so we have to find more images for other classes.

Adding more convolutional layers to the model improves the accuracy of the test set. As a result, we can propose a deeper neural network with more convolutional layers and other advanced layers to improve the system. Namely, TensorFlow does not natively support spatial transformation networks for this experiment and implementing them will surely yield better results.

6 Conclusion

From the experiments that we have run, the best model that we came up with is the one with four convolutional layers of 32, 64, 128, and 256 filters, a drop rate of 0.2, dense layers of 256, 128, 43 neurons and using the adam optimizer. Using this architecture, while we were unable to reproduce the results of the state-of-the-art model, we were achieved a 95.01% accuracy for our test.

References

[1] Álvaro Arcos-García, Juan A Alvarez-Garcia, and Luis M Soria-Morillo. Deep neural network for traffic sign recognition systems: An analysis of spatial transformers and stochastic optimisation methods. *Neural Networks*, 99:158–165, 2018.

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[3] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE International Joint Conference on Neural Networks*, pages 1453–1460, 2011.

[4] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012.