

Ensemble Methods to Build Next-Generation Traffic Sign Recognition System for Self-driving Car

Yifei Pei

Dpt. of Computer Engineering
Santa Clara University,
Santa Clara, Ca, 95053
ypei@scu.edu

Andrew Song

Dpt. of Computer Engineering
Santa Clara University,
Santa Clara, Ca, 95053
asong@scu.edu

Abstract

In this project, we have worked on an image recognition system using convolutional neural networks with ensemble method for autonomous cars. Before training the model, we used preprocessing methods like grayscaling the images and using histogram equalization on them. We then made the system using the LeNet architecture, a neural network architecture which was primarily used for recognizing grayscaled handwritten digits. However, the testing accuracy wasn't high enough, so we had to add some convolutional layers on them as well as an ensemble method to increase the test accuracy and to solve the overfitting problem. We also added an image generator to solve the overfitting problem as well. Overall, with the right number of correct models to be bagged, we have significantly improved the testing accuracy for such a small dataset, which consisted of images of German traffic signs. In fact, we also have found that there is a positive relationship between the number of bagged models with the testing, training, and validation accuracies.

Keywords: Image Classification; Convolutional Neural Network; Ensemble Methods; Bagging

Introduction

Autonomous vehicles have been an active area of research for the last few decades. Intensive research has been done on using a front viewing camera for vehicle localization and navigation, environment mapping, and obstacle avoidance. Accuracy is the most basic requirement and evaluation metric for the traffic sign recognition algorithms.

One of the important tasks for self-driving cars is image recognition. The cars need to accurately recognize the images and take appropriate action based on the image and the data inputted in the car (e.g. GPS). Our goal is to combine different methods and models (ensemble learning) to build a strong image classifier to recognize traffic signs.

Methods

Theories

There are three major types of traffic sign detection methods that have been intensively studied in literature. These three types of methods are the following: color-based methods, shape-based methods, and learning based methods.

In the paper “Traffic Sign Detection for Vision-based Driver’s Assistance in Land-based Vehicles” [1], Varun introduced a threshold method to detect the color red in the RGB color space [2], which consists of three color planes: red, blue, and green. Besides color detection, shape-based detection is another category of detection methods, which is based on shape recognition. In [3] and [4], the authors used the Hough transform of the edge image and detected the shape in the image solely based on the angle separations. In [5], a gradient-based centroid voting scheme was developed to find the centroid of approximate shapes in a gradient image. Finally, for learning-based detection methods, traditional traffic sign detection methods usually require some a priori knowledge of the color and shape of the traffic sign. In recent years, machine learning techniques like support vector machine and convolutional neural network have been studied and implemented as detection algorithms.

Germany has a comprehensive and uniform traffic sign system [6]. All signs have standard shapes and colors and use international pictograms and symbols based on the Vienna Convention on Road Signs and Signals, which are easy to understand. The sign system had its last complete overhaul in the early '90s to closely conform more with European standards and is updated every few years to address any changing needs and shortcomings.

LeNet-5 (LeNet), a pioneering 7-level convolutional network by LeCun et al. created in 1998 that classifies digits, was applied by several banks to recognize handwritten numbers on checks digitized in 32x32 pixel images [7]. The ability to process images of higher resolution requires larger and more layers of a convolutional neural network, so this technique is constrained by the availability of computing resources. However, given that the GPU’s computing power is high [8], we can come up with our modified model based on LeNet. The reason to use LeNet as our base model is that one can easily build and tune the hyperparameters. A hyperparameter is a parameter of a prior distribution; the term is used to distinguish them from parameters of the model for the underlying system under analysis. In a convolutional neural network, hyperparameters indicate the features of filters (kernels) of convolutional layers and pooling layers.

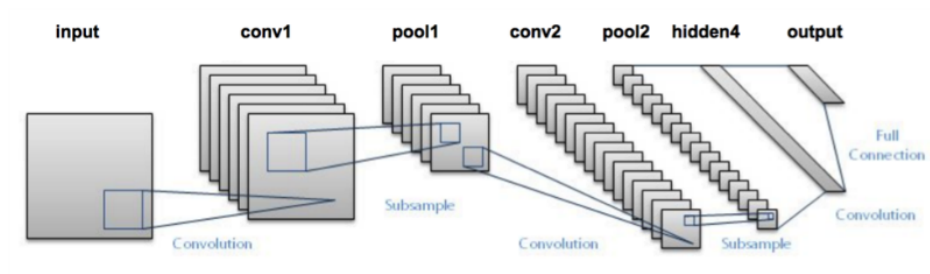


Figure 1: LeNet Architecture.

In practice, the neural network has a possibility of overfitting. When overfitting occurs, the model fits the training dataset well but cannot fit the new dataset well. One way to detect overfitting is to use the model on a small dataset called validation dataset. If the accuracy of the training dataset is higher than that of the validation dataset, the model is prone to overfit data. Some possible ways to correct overfitting include using more data to train the model and add regularization terms. In neural networks, the most widely used method is to add dropout layers^[9]. Dropout layers randomly drop some nodes in the neural network, which could prevent overfitting.

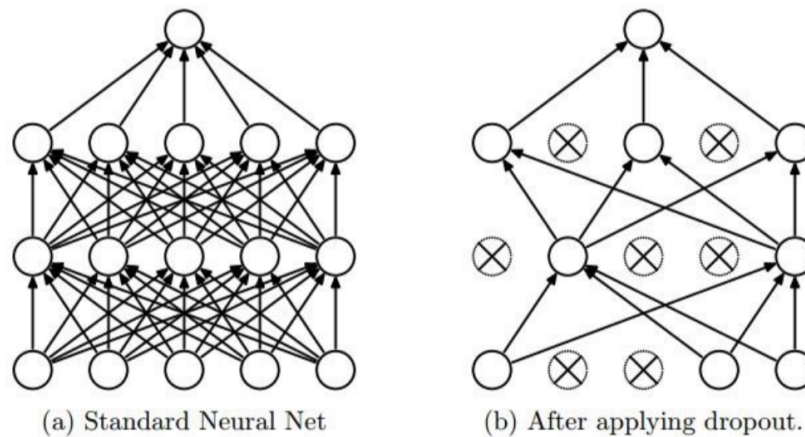


Figure 2: Neural net before and after applying dropout.

Ensemble methods are techniques that create multiple models and then combine them to produce improved results^[10]. Ensemble methods usually produce more accurate solutions than a single model does. However, because ensemble methods require high computing power, they were not frequently used until the recent decade. In the popular Netflix Competition, the winner used an ensemble method to implement a powerful collaborative filtering algorithm. Another example is KDD 2009 where the winner also used ensemble methods. The most popular techniques of ensemble methods include bootstrap or bagging and boosting. Boosting combines weak machine learning models to create a stronger one. However, it's not commonly used in a neural network because the neural network is usually classified as a strong model. However, bootstrap (often called bagging) can be applied to any machine learning models. The first step of bagging is to resample training dataset for n times and get n sample subsets. The second step of bagging is to resample training datasets multiple times to train one model individually to get multiple models^[11]. The final step combines the results by these models. For the regression problem, we use the mean of output predicted by the models as the result. For the classification problem, we use mode of the results by vote. The bagging method increases the bias but reduces the variance much more. Thus, bagging has the potential to reduce the error of prediction and improve the accuracy of machine learning models.

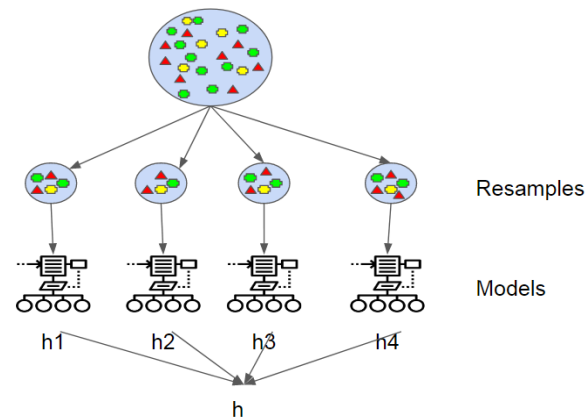


Figure 3: Bagging.

Histogram equalization is a method that usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values ^[12]. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values. Histogram equalization can enhance the contrast of images, which will improve image classification models.

In our project, we build our model based on the LeNet since it's easy for us to tune the hyperparameters. We add dropout layers and convolutional layers to improve our models. Considering that the shapes of signs in the images are dominant in our image classifiers and we only have free Google GPU, which is in Google Colaboratory available online but has limited runtime, we converted images from RGB (3 color channels for red, green and blue) format to grayscale. To make our training process more accurate, we enhanced the contrast of images by histogram equalization. Because our training dataset is not large, we used Keras image generator module to generate new images randomly, which can be used in our training process to get our optimal model. Finally, we incorporated bagging methods to improve our model to build a strong traffic sign classifier.

Exploratory Data Analysis (EDA)

Before we built our traffic sign classifier model, we needed to inspect our dataset. We plotted our training data into a histogram. We found that there were 43 classes of traffic sign images in our training dataset. Some classes appeared more frequently than others, which is normal in our life. A dataset name 'signname.csv' showing the sign names for the 43 classes of traffic signs. We randomly selected 5 images of each class and show the first 5 classes in Figure 5.



Figure 4: EDA Results.



Figure 5: Some of the classes with corresponding images.

Grayscale Transformation and Histogram Equalization

We converted images in training dataset, validation dataset, and testing dataset from RGB format to grayscale format. Then, we applied histogram equalization to every image. We used OpenCV2 to complete our image processing. We show 5 images with their transformed images in the chart. One can see that the an important feature of the sign is the information (e.g. arrows) and histogram equalization enhances the feature in the image. We also found that the quality of histogram-equalized images depended on that of the respective original images.




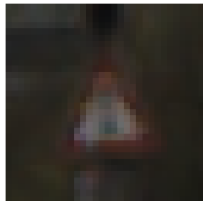
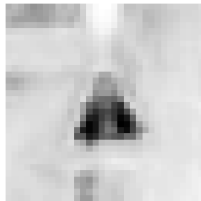
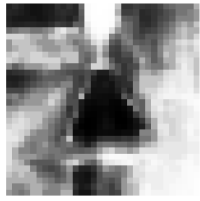






Image Number	Original	Grayscale	Histogram Equalization
500			
1500			
2500			
5000			

Figure 6: Images, color, grayscale, and histogram-equalized.

Initial Model Using LeNet with Dropout Regularization

We built our LeNet model with one dropout regularization. The model summary is shown in the chart below. In the first convolution layer, we have included 60 filters of 5x5 pixels, filtering out the 32x32x1 images that came through, resulting in the images with the dimensions of 28x28x1. In the second one, we included 30 filters of 3x3 pixels on images that have already had their dimensions halved from the pooling layer. The max pooling layers (2x2) were used to downsample the pixels of images. All the activation functions in the layers except for the output layer are ReLu functions. The activation function for the output layer is the softmax function, which converts the outputs into the probabilities. For certain images, because there are 43 output nodes for 43 classes, the neural network generates 43 outputs and converts them into 43 probabilities. one can tell which class the image belongs to by finding the highest probability for that image. After training our LeNet neural network and testing the validation dataset and testing

dataset, we found that the training accuracy is higher than the validation accuracy (0.9817 vs. 0.9381), which indicates the model is overfitting. The test accuracy is not high (0.9249). The accuracy can bring cause the car to run into trouble if model is applied to self-driving car traffic sign classifier systems.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 28, 28, 60)	1560
max_pooling2d_1 (MaxPooling2)	(None, 14, 14, 60)	0
conv2d_2 (Conv2D)	(None, 12, 12, 30)	16230
max_pooling2d_2 (MaxPooling2)	(None, 6, 6, 30)	0
flatten_1 (Flatten)	(None, 1080)	0
dense_1 (Dense)	(None, 500)	540500
dropout_1 (Dropout)	(None, 500)	0
dense_2 (Dense)	(None, 43)	21543
Total params: 579,833		
Trainable params: 579,833		
Non-trainable params: 0		
None		

Figure 7: Initial Model Summary for LeNet

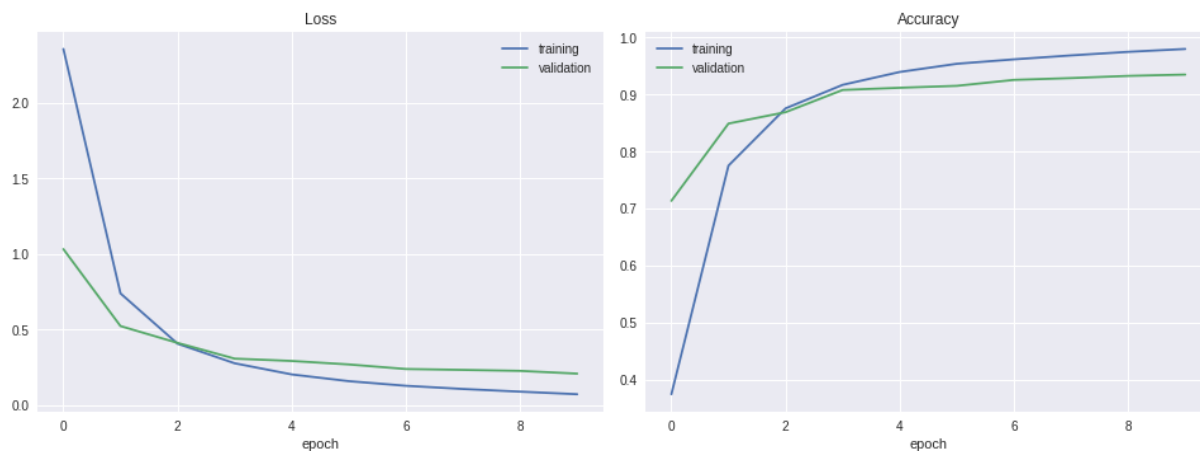


Figure 8: LeNet model's results.

Modified Model with Additional Convolutional Layers

Given what we got, we doubled the convolutional layers. Because the number of parameters has increased, we saw that our training process has slowed down. The training accuracy was 0.9869, which is slightly higher than validation accuracy (0.9732). But overfitting still existed, and the testing accuracy was still low (0.9467).

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 28, 28, 60)	1560
conv2d_4 (Conv2D)	(None, 24, 24, 60)	90060
max_pooling2d_3 (MaxPooling2D)	(None, 12, 12, 60)	0
conv2d_5 (Conv2D)	(None, 10, 10, 30)	16230
conv2d_6 (Conv2D)	(None, 8, 8, 30)	8130
max_pooling2d_4 (MaxPooling2D)	(None, 4, 4, 30)	0
flatten_2 (Flatten)	(None, 480)	0
dense_3 (Dense)	(None, 500)	240500
dropout_2 (Dropout)	(None, 500)	0
dense_4 (Dense)	(None, 43)	21543
Total params: 378,023		
Trainable params: 378,023		
Non-trainable params: 0		
None		

Figure 9: Modified model summary.

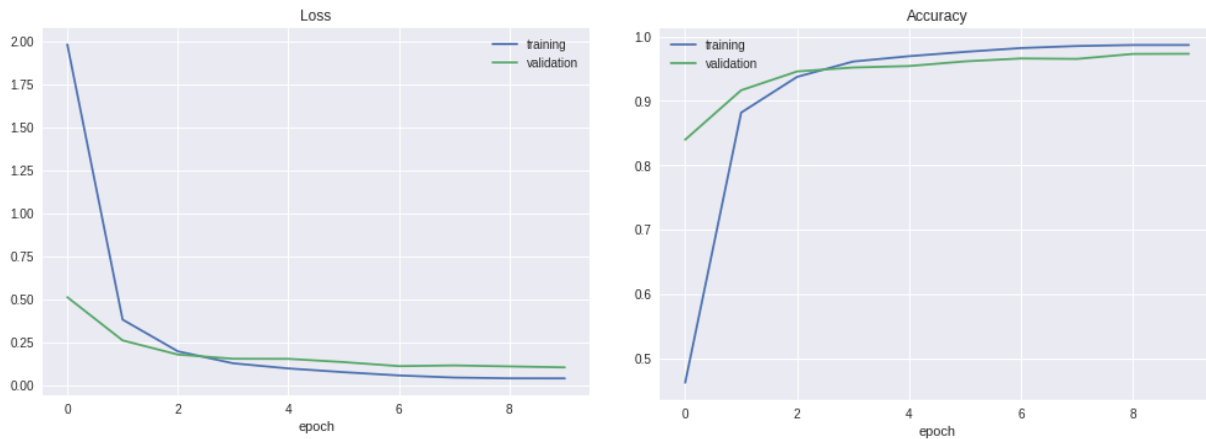


Figure 10: Modified model's result.

Image Generator

Keras has image generator that can randomly generate new images based on our training dataset. The parameters in the generator includes width shift range, height shift range, zoom range, shear range and rotation range.

We set the parameters as below:

“width_shift_range = 0.1” indicates the 10% maximum horizontal shift;

“height_shift_range = 0.1” indicates the 10% maximum vertical shift;

“zoom_range = 0.2” means that one can zoom in or out at the maximum of 20%;

“shear_range = 0.1” means that the image can shear towards the screen at the max of 10%;

“rotation_range = 10” means that the image can rotate about its center point to a max of 10

degrees.

For each training epoch (total of 10), we used image generator 2000 times. For each time, the image generator generated 50 new images based on our training dataset randomly. The training processes are based on the new images, which are 100,000 images for each epoch. We randomly picked 10 new images generated by Keras image generator and show them below.



Figure 11: Generated images.

The training accuracy was 0.9826, the validation accuracy was 0.9952, and the testing accuracy was 0.9794. We can see the validation accuracy is higher than the training accuracy, which indicates that we solved the overfitting problem. We also found that the testing accuracy has made a huge increase.

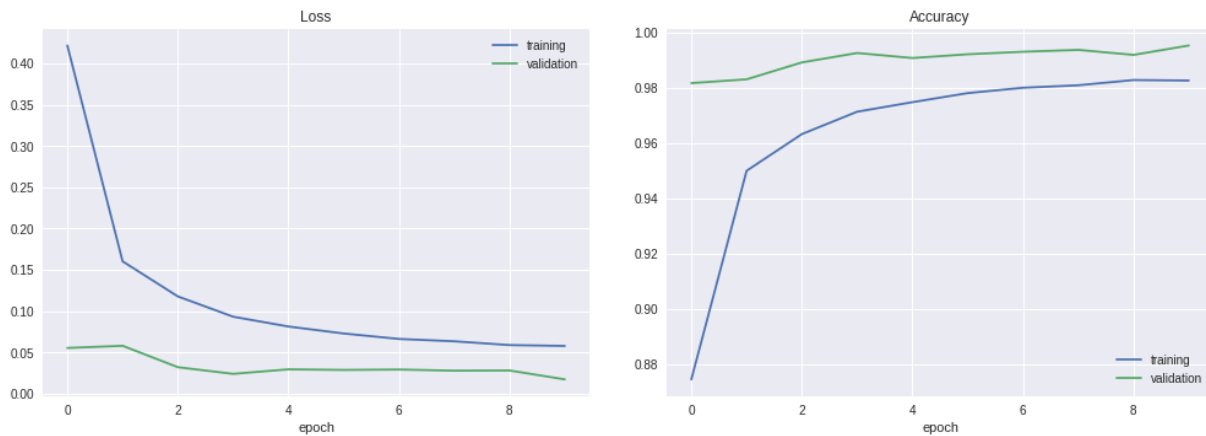


Figure 12: Image Generator Model's Result.

Bootstrap (Bagging)

For our project, we used 10% of our training dataset to resample 10 times to get 10 sample subsets to train our model and ensemble them to predict our testing dataset. Then, we used all the training datasets to resample. For this experiment, we set $n = 3, 5, \text{ and } 10$ independently.

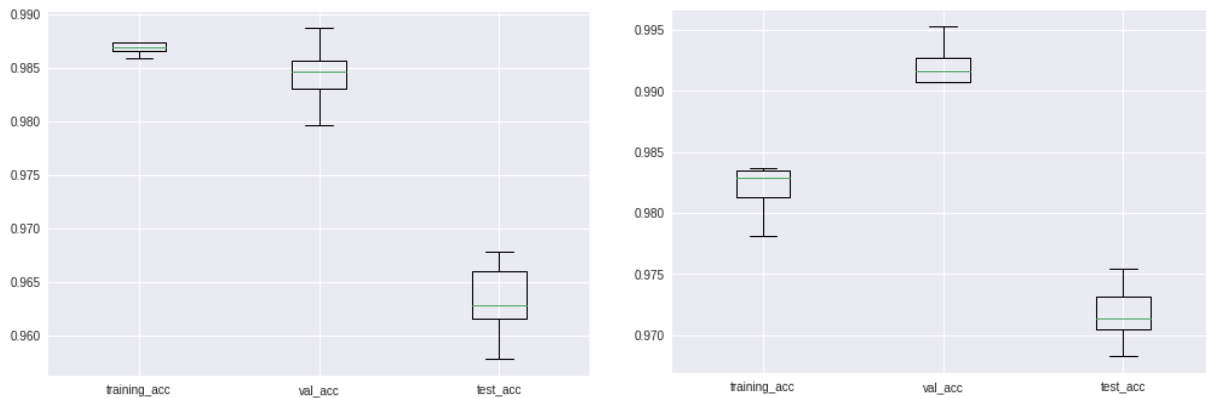


Figure 13: Boxplots for Accuracies of 10 individual models with 10% Training Data (Left) & 100% Training Data (Right).

Bagging 10 individual Models Results:

10% Training Data (10 Models):

Testing Accuracy of Bagging Model : 0.9836

Training Accuracy: 0.9982

Validation Accuracy: 0.9945

100% Training Data (10 Models)

Testing Accuracy of Bagging Model : 0.9866

Training Accuracy: 0.9995

Validation Accuracy: 0.9972

We observed that the validation accuracies are lower than the training accuracies in the 10 individual models with 10% of the training data, which indicates that individual models with fewer training dataset tend to be overfitted. However, after we ensemble these 10 individual models using bagging method, we found that the validation accuracy (0.9945) is almost same as training accuracy (0.9982) and testing accuracy jumped by about 0.02, resulting a high testing accuracy (0.9836). We also find that the testing accuracy (0.9866) in the 10-ensemble models with 100% training data is slightly higher than the testing accuracy (0.9836) in the 10-ensemble models with 10% training data, and only jumped by about 0.03.

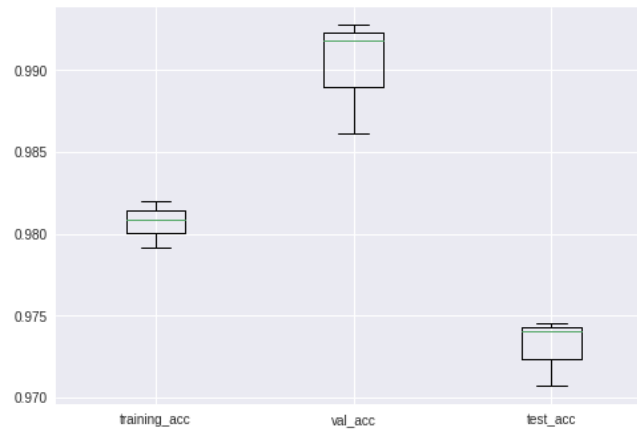


Figure 14 : Box plots for accuracies of 3 individual models with 100% training data.

Bagging 3 Individual Models with 100% Training Data Results:

100% Training Data (3 Models):

Testing Accuracy of Bagging Model : 0.9795

Training Accuracy: 0.99928

Validation Accuracy: 0.9994

We observed that without bagging, individual models with 100% training data did not have the overfitting problem. However, the validation accuracy (0.9941) in the 3-ensemble model was lower than training accuracy (0.99928) in that model, which indicates that the 3-ensemble model had the overfitting problem. Nevertheless, bagging improves individual models' accuracies.

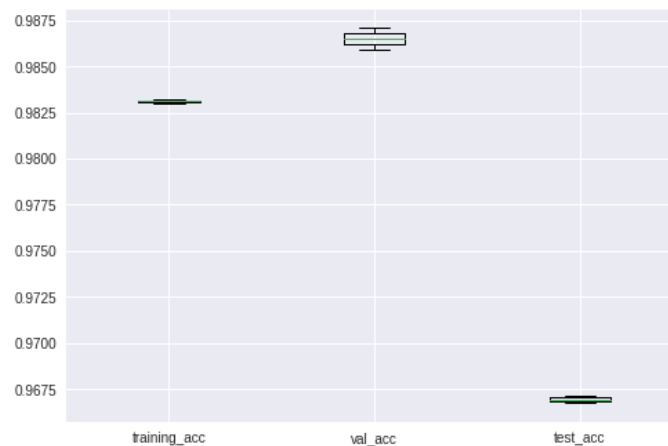


Figure 15: Boxplots for Accuracies of 2 individual models (results randomly picked when encountered) with 100% training data.

Bagging 2 Individual Models with 100% Training Data Results:

100% Training Data (2 Models):

Testing Accuracy of Bagging Model : 0.9639

Training Accuracy: 0.9958

Validation Accuracy: 0.9836

Bagging two individual models with 100% training data did not have a huge impact on validation

accuracies and testing accuracies. However, doing so greatly improved training accuracies, which can lead to overfitting.

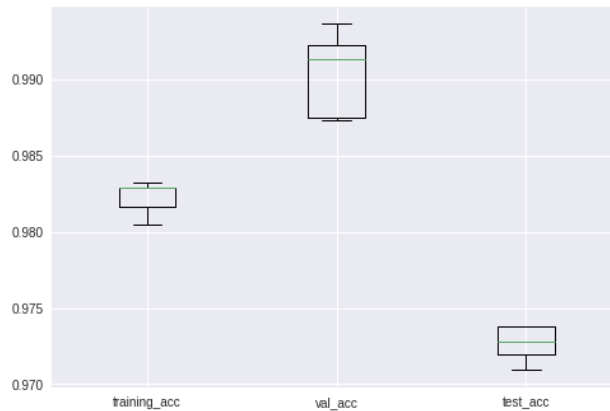


Figure 16: Boxplots for accuracies of 5 individual models with 100% Training Data

Bagging 5 Individual Models with 100% Training Data Results:

100% Training Data (5 Models):

Testing Accuracy of Bagging Model : 0.9827

Training Accuracy: 0.9995

Validation Accuracy: 0.9956

Bagging 5 individual models with 100% training data yielded the high testing accuracy (0.9827) with the validation accuracy (0.9956) almost equal to the training accuracy (0.9995).

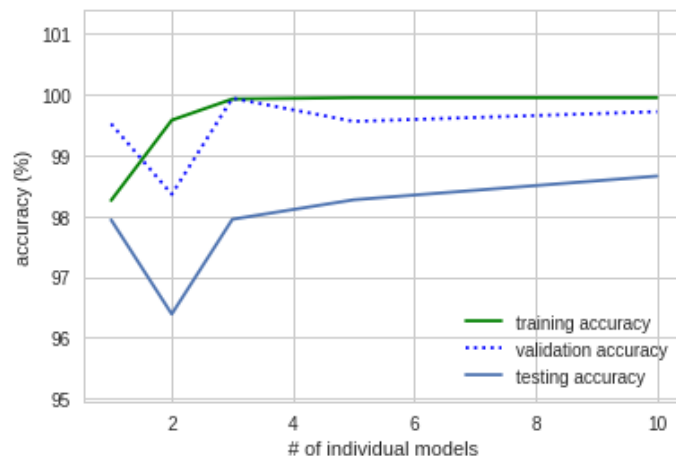


Figure 17: The relationship between accuracies and the number of training models.

Discussions and Conclusions

We combined several methods to improve LeNet model and get good results. We found that convolutional neural network has potential overfitting problems. However, the image generator and ensemble method, bagging, were used to solve overfitting issues. Nevertheless, we had to consider the number of individual models when we use bagging. As we observed from our results, bagging a small number of individual models may not help the overall model and yet it

may cause overfitting. Typically, bagging more individual models yields better results because bagging reduces variance among the prediction results, as shown in Figure 17. However, we also had to consider whether the individual models were suitable for bagging. In our bagging model, we combined several models that have been improved. However, if we had combine the models that do not fit to the dataset, we would not have gotten good results.

References

- [1] S. Zhang. "Traffic Sign Detection for Vision-based Driver's Assistance in Land-based Vehicles". pp. 1.
- [2] S. Varun, S. Singh, R. S. Kunte, R. D. S. Samuel and B. Philip, "A Road Traffic Signal Recognition System Based on Template Matching Employing Tree Classifier," *International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007)*, Sivakasi, Tamil Nadu, 2007, pp. 360-365.
- [3] v. Barrile, M. Cacciola, G. M. Meduri, and F. C. Morabito. "Automatic Recognition of Road Signs by Hough Transform: Road-GIS." *emphJournal of Earth Science and Engineering*, vol. 2, no. 1, 2012.
- [4] M. A. Garca-Garrido, M. . Sotelo, and E. Martn-Gorostiza. "Fast road sign detection using Hough transform for assisted driving of road vehicles." *International Conference on Computer Aided Systems Theory*. Springer Berlin Heidelberg, 2005.
- [5] G. Loy, Fast shape-based road sign detection for a driver assistance system, in In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2004, pp. 7075.
- [6] <http://www.gettingaroundgermany.info/zeichen.shtml>
- [7] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.
- [8] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone and J. C. Phillips, "GPU Computing," in *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879-899, May 2008.
- [9] ByungSoo Ko, H. Kim, Kyo-Joong Oh and H. Choi, "Controlled dropout: A different approach to using dropout on deep neural network," *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Jeju, 2017, pp. 358-362.
- [10] N. Tabassum and T. Ahmed, "A theoretical study on classifier ensemble methods and its applications," *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, 2016, pp. 374-378.

[11] N. Tabassum and T. Ahmed, "A theoretical study on classifier ensemble methods and its applications," *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, New Delhi, 2016, pp. 374-378.

[12] W. Zhihong and X. Xiaohong, "Study on Histogram Equalization," *2011 2nd International Symposium on Intelligence Information Processing and Trusted Computing*, Hubei, 2011, pp. 177-179.