Traffic Sign Image Classification Using Convolution Neural Network and Computer vision

Sai Amarnath Chintha
dept. of computer science
Georgia State University
Atlanta, USA
schintha1@student.gsu.edu

Dhara Mandal
dept. of computer science
Georgia State University
Atlanta, USA
dmandal1@student.gsu.edu

Dr. Yanqing Zhang
Professor, *dept. of computer science*
Georgia State University
Atlanta, USA
yzhang@cs.gsu.edu

***Abstract*—Self-Driving Car research problem requires several sub-topics that need to be discussed more deeply. Such as Deep learning, Computer Vision, Fusion Sensor, Localization, Control, until Path Planning. All of them are fusion of several fields of study. Recently, many researchers and tech companies are competing to develop self-driving car using different approaches. There are many CNN methods that aims to detect objects from images and classify them. On the other hand, the advancement of Computer Vision these days has grown up beyond imagination. Road Lane Detector is used to detect road track that can be helpful for the decision-making process of the self-driving car. In this paper we tried to optimize lenet convolutional neural networks to classify the road sign images.**

Keywords—CNN (Convolution Neural Network), classification, Data preprocessing, Data mining techniques, Optimization.

I. INTRODUCTION

In recent years, the autonomous vehicle technology has been the center of attraction for researchers from academia and the industry-alike. Various endeavors and experiments have been performed, which demonstrated advancements in the detection and recognition of traffic signs. Different techniques have been adopted for this task and were discussed in [1]. Traffic sign recognition is challenging due to the fast-changing environment and illumination, cluttered background, low resolution, shape distortion resulted from changing camera angles.

In this paper we start with Lenet Convolutional neural networks and further optimize to have better accuracy and high computational power than using a traditional neural network and generalize a solution to classify Traffic sign images. As, for all the traffic sign detection techniques we use training images to train the model. So, we propose methods for preprocessing and improving the accuracy in our case of Traffic sign detection. The dataset we are going to use to train out model is ununiform and much more complex dataset to train our model. We will go through a series of preprocessing techniques and improvement methods to get good training and testing accuracy.

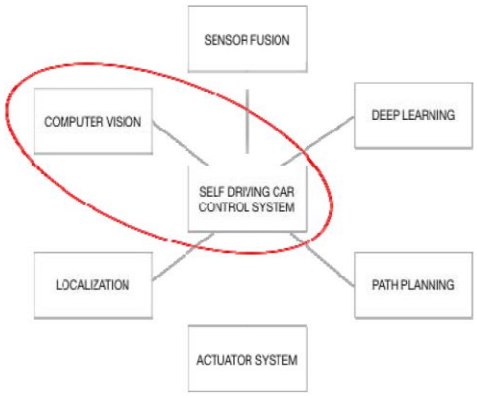
II. PREVIOUS RESEARCH

To recognize traffic signs, some researchers employed a pixel-based approach and determined the class of traffic signs using cross-correlation template matching or neural networks or convolutional neural network (CNN). Pixel-based template matching cannot yield high performance because of its sensitivity to gray-scale variation and shape deformation. The CNN can learn discriminative features from pixels in a hierarchy of layers to lead to high accuracy of classification [2], but both training and classification suffer from high computation complexity because of the large number of connections between neurons. Many papers have been published about it, especially in Deep Learning and Convolutional Neural Network (CNN). The CNN method has largely been exploited in various fields: image processing, machine learning, video analysis, natural language processing, and much more.

GoogleNet [3], the winner of ILSVRC 2014, uses a deep CNN architecture with 22 network layers which are called as Inception. The core of the architecture is by piling convolutional and pooling layer in a sequential manner. With this high and piling computation demands, the computation time will be its downside where the detection approach that we want to use is in a real-time environment. Faster R-CNN [4], one of the incremental invoked CNN method that takes advantage of the convolutional network sharing from Fast RCNN [5], uses Region Proposal Network (RPN) to generate the region of object proposals which will be used by Fast R-CNN for object detection. Even though they already use CNN method in all the steps to speed-up the computation time, it still could achieve up to 7 frames-per-second (fps). Based on our needs for real-time object detection, these previous two methods are still not applicable for that. SSD [4], on the other hand, tackles the problem, and the overall results are better on the YOLO version 1 [6].

III. CONCEPT

Initially we present an overview of the technologies used in a Self-Driving car. Self-Driving car is build using a wide range of technologies such as Computer Vision, Sensor Fusion, Deep Learning, Path Planning, Actuator Localization as figure



2

Fig.1 Block diagram of Self Driving Car System

A. Computer Vision

First, The basics of computer vision start with defining an image as an image to humans is extremely complex. It

involves color, contrast, shading, focus, and many other factors. As humans, we see these elements as one, yet we still analyze a picture based on those specific features. Color and contrast helps to characterize an object and to define an object's bounds, and shading and focus gives three-dimensional perspective. So it makes sense to try to map a human's vision system to computer code, as we are most familiar with our system. Maintaining the Integrity of the Specifications

B. *Sensor Fusion*

Sensor fusion is the aggregation of data from multiple sensors to gain a more accurate picture of the sensors' subject or environment than can be determined by any one sensor alone. Also, sensor fusion is combining of sensory data or data derived from disparate sources such that the resulting information has less uncertainty than would be possible when these sources were used individually.

C. *Deep Learning*

Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be classified as supervised, semi-supervised or unsupervised. Also, with the help of deep learning salient features can be automatically detected and processed by using deep learning algorithm

D. *Path Planning*

Path planning is used to find the shortest or optimal path between two points. This is very important primitive as it could minimize the amount of turning, the amount of braking or whatever a specific application requires.

E. *Actuator System*

Actuators are the components of a machine responsible for controlling and moving the system. Actuators are like muscles of your body, responding to electrochemical signals from your brain so that you move such parts as arm or leg.

F. *Localization*

Robot navigation means the robot's ability to determine its own position in its frame of reference and then to plan a path towards some goal location. To navigate in its environment, the robot or any other mobility device requires representation

IV. Background

The traffic sign recognition systems are usually comprised of two parts: traffic sign detection and classification. The regions of interest are first extracted in detection, and then identified correctly or rejected in classification. Usually, detection is performed by a classifier via sliding windows [3]. Researchers have employed various approaches to detect and extract the features of interest from input images and classify them.

V. Proposed Model

Here we focused to understand how the CNN model has been employed in traffic sign classification.

A. *Dataset and Preprocessing*

The dataset contains more complex, multidimensional and relevant images. The dataset is relatively smaller if compared with the images present when compared to training set, validation set and the test set. This dataset has 43 classes to classify images. Not only our dataset is smaller we also have fewer images belongs to each class Since the dataset is being classified into many more classes. It might make a lot more difficult to our model to learn the accuracy with the high property rate.

Importing the dataset: Importing the dataset, which is a directory from bit bucket by cloning from a repository.

Dataset Name: German-traffic-signs. It has four files in it: signmanes.csv (a csv files that contains all of our traffic signs along with the labels), test.p, train.p, valid.p rest of them are testing, training and validation and are pickle files. Next step is loading and unpickling the files. Every file in the dataset is in dictionary format label as a key value. Our training file has 34799 images of 32*32 pixels with a death of 3, which means they are in RGB format. So as the validation dataset has 4410 images and testing set has 12630 images.

```
(34799, 32, 32, 3)
(12630, 32, 32, 3)
(4410, 32, 32, 3)
```

Fig.2 Summary of Dataset

For our convolutional neural network to get trained we change our training set into format of a grid with each class in row

Fig.3 Grid representation of Dataset

When we print the distribution of our training dataset we can see that it's ununiform with as less as 180 images to 2010 images to train the model. So, we get less images for some classes to train compared with others.

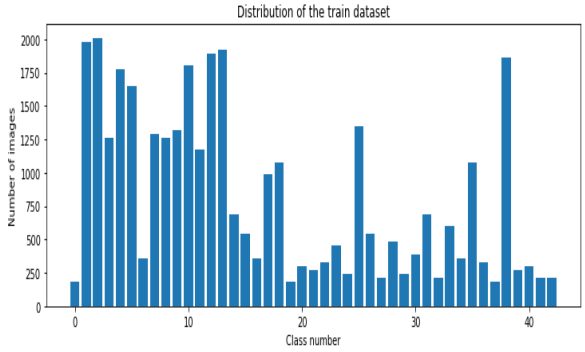


Fig.4 Distribution of Training Dataset

Our traffic sign dataset is complex with RGB format and have a variety of background. And each one of them have a unique extra feature making more difficult to classify them. Therefore, will preprocess them to make easier for our model to classify them.

Preprocessing techniques proposed to ensure high accuracy:

Converting RGB into grayscale: First step is to convert RGB images into grey scale. This conversion is important for many reasons. To identify traffic signs, color is not a very significant feature to look for. Moreover, lighting in our images varies and each one of them have different colors. So, color seems to be not a relevant feature to look for in our case. Also, when we convert from 3-dimensional to one, we have less input parameters to give hence it reduces a lot of computational power to train. On the other hand, the more important feature to look for are edges, shape and the curves inside an image.

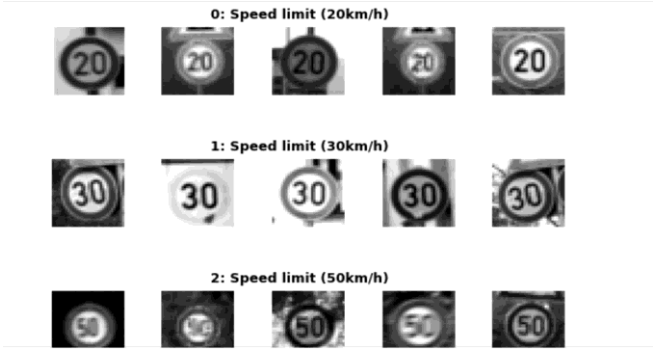


Fig.5 Greyscale converted images

Histogram equalization: The goal of histogram equalization is to standardize the lightening effect in all our images. Some of our image are very bright and the other are dim. In this case after histogram equalization all our images have a similar lightening effect. If an image has the pixel intensities spread over the smaller range of brightness values histogram equalization takes our histogram and spreads it over the ends that covers the higher range of brightness values and helps to normalize, lightening all our image. This process also results in a higher contrast with in our images which helps the feature extraction. So now all grey scale images are better distributed across the image and deemphasizing the images that occur at high frequency. This can be explained theoretically by flattening the brightness values.

$$H'(i) = \sum_{0 \le j < i} H(j)$$

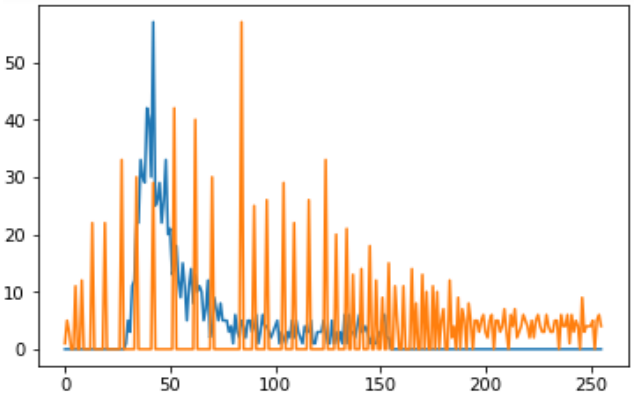


Fig.6 Histogram Equalization example

Normalization: Here we divide all our picture pixels by 255. This causes all our pixel value to be normalized between zero and one. Data normalization is an important step which ensures that each input parameter (pixel, in this case) has a similar data distribution. This makes convergence faster while training the network. Data normalization is done by subtracting the mean from each pixel, and then dividing the result by the standard deviation. The distribution of such data would resemble a Gaussian curve centered at zero. For image inputs we need the pixel numbers to be positive, so we might choose to scale the normalized data in the range [0,1] or [0, 255].

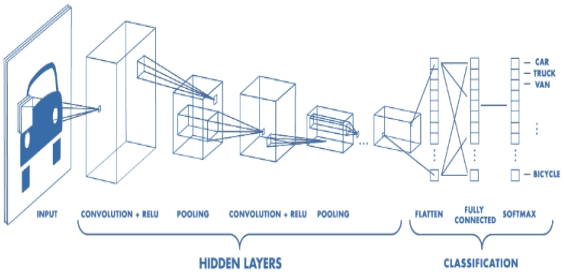
Now we have our data ready to be used by our convolutional neural network

B. Convolution Neural Network

Convolutional Neural Networks have a different architecture than regular Neural Networks. Regular Neural Networks transform an input by putting it through a series of hidden layers. Every layer is made up of a set of neurons, where each layer is fully connected to all neurons in the layer before. Finally, there is a last fully-connected layer—the output layer—that represent the predictions. Convolutional Neural Networks are a bit different. First of all, the layers are organized` in 3 dimensions: width, height and depth as shown in fig 4. Further, the neurons in one layer do not connect to all the neurons in the next layer but only to a small region of it. Lastly, the final output will be reduced to a single vector of probability scores, organized along the depth dimension.

C. Feature Extraction and Reduction using Lenet Convolutional Neural Network

Using **lenet** model convolutional neural network to test the results: we first add a convolutional layer to our model starting with 30 filters inside the convolutional layer and use “relu” as our activation function. The output of this function is (28,28,30). Our (32,32,1) image is being reduced to 28*28 with a death of 30. Since we are using 30 independent filters. The model has a total of 780 parameters to learn. Next, we add a pooling layer as per the architecture of the lnet model. The pooling size we give is 2*2. So now our image size is reduced by half of its original size and death still be the same. And this death contains all the important feature values which is very valuable. We now add other convolutional layer with 15 filters with 3*3 size. Which leads to 4065 parameters. The output is going to have a death of 15. As we are going to take our convoluted data and going to fed into the fully connected layer as a one-D array we use flatten. Therefore 540 nodes will be fed into our multi-layer perceptron. We add dense layer with 500 nodes to fit the data well arbitrable. For the output layer we use 43 nodes and SoftMax as our activation function.



Architecture of Lenet CNN

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
flatten_1 (Flatten)	(None, 480)	0
dense_1 (Dense)	(None, 500)	240500
dropout_1 (Dropout)	(None, 500)	0
dense_2 (Dense)	(None, 43)	21543

Fig.7 Summary of Proposed Convolutional Neural Network

```
Total params: 378,023
Trainable params: 378,023
Non-trainable params: 0
```

We train the model using our training data and set epoch to 10 and validate using validation data. At the end of the 10 epoch the training accuracy seems to be 97, The validation accuracy initially started with higher than the training accuracy but ended with 93

.

Epoch	*Val_loss*	*Val_accuracy*	*Loss*	*Accuracy*
1	0.1048	0.9635	0.9481	0.7214
2	0.0608	0.9819	0.2450	0.9239
3	0.0408	0.9898	0.1620	0.9498
4	0.0457	0.9839	0.1294	0.9593
5	0.0399	0.9887	0.1069	0.9675
6	0.0370	0.9898	0.0944	0.9706
7	0.0268	0.9918	0.0827	0.9744
8	0.0436	0.9860	0.0768	0.9767
9	0.0271	0.9921	0.0746	0.9778
10	0.0261	0.9923	0.0671	0.9793

Fig.8 Accuracy vs Epoch

Evaluating our model

```
Test Score: 0.42523097056815856
Test Accuracy: 0.9082343625909076
```

The performance of our model is not very good. Evaluating the performance, we see the loss is even higher and a minimum value and validation loss is even higher of about 0.25

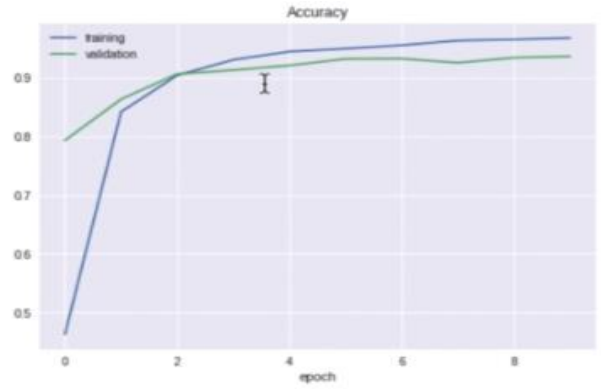


Fig.9 Accuracy vs Epoch

we see that a training accuracy about 0.96 while our validation accuracy lags with a maximum of 0.93

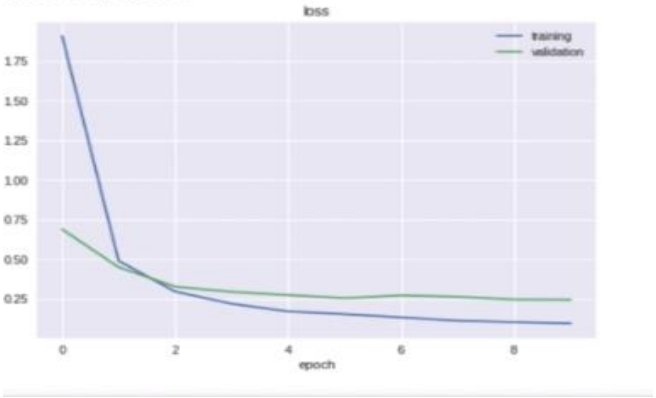


Fig.10 Loss vs Epoch

These values imply that our network is not performing effectively in terms of predicting images of our dataset. And seems that our network has overfitted data. This can be said by seeing our graphs as validation accuracy is less than training accuracy.

D. Fine Tuning our model

Decrease Learning Rate: A learning late that is too high can often leads to low accuracy. If we give a low learning rate our network model can learn more effectively when a more complex dataset is involved. So, we decrease of learning to 0.001. when we train our model using modifies learning rate our accuracy is increased. However, our model is still overfitting.

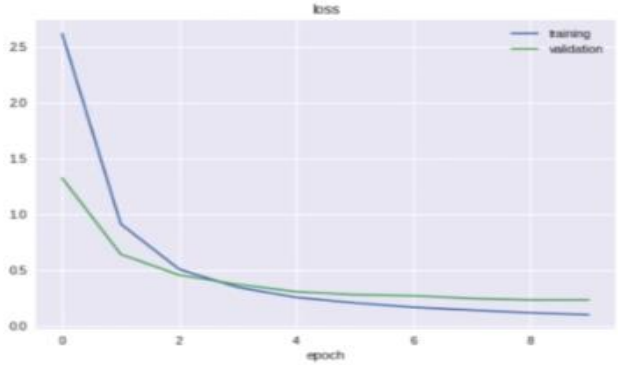


Fig.11 Loss vs Epoch in Learning rate modified version

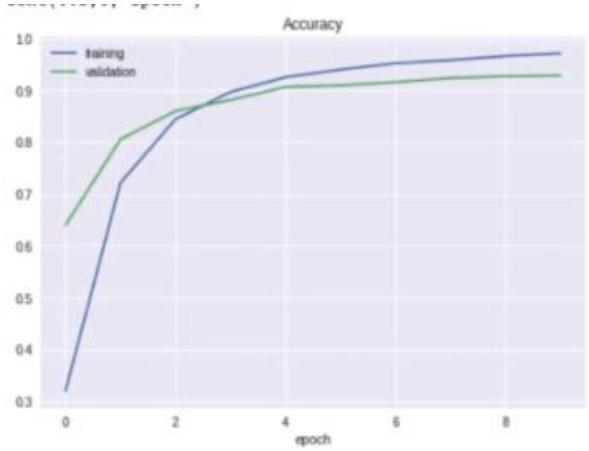


Fig.12 Accuracy vs Epoch in Learning rate modified version

```
Test Score: 0.34709213727160476
Test Accuracy: 0.9113222486049716
```

Increasing the number of filters: Increasing the number of filters in our CNN can help our network to extract more number of features which results in improving the accuracy. We double the number of filters than defined before. Now this changes all our parameters. Now when we train our model at the end of the 10th epoch we have an accuracy of 0.9809 and validation accuracy of 0.9347.

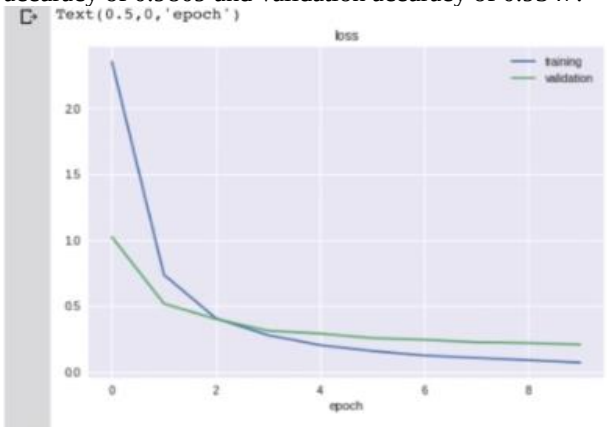


Fig.13 Loss vs Epoch in Filter modified version

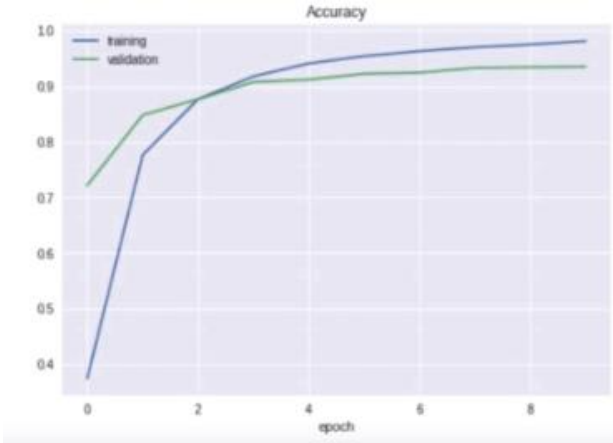


Fig.14 Accuracy vs Epoch in Filter modified version

```
Test Score: 0.30612055453769005
Test Accuracy: 0.9250197941598114
```

Adding an Extra convolutional layers: Now we try by adding an extra convolutional layer to our network. We will use same amount and size of filters and same activation function.

conv2d_23 (Conv2D)	(None, 28, 28, 60)	1560
conv2d_24 (Conv2D)	(None, 24, 24, 60)	90060
max_pooling2d_15 (MaxPooling	(None, 12, 12, 60)	0
conv2d_25 (Conv2D)	(None, 10, 10, 30)	16230
conv2d_26 (Conv2D)	(None, 8, 8, 30)	8130
max_pooling2d_16 (MaxPooling	(None, 4, 4, 30)	0

Fig.15 Extra Convolutional layer

When we train our model now and test for accuracy:

```
Test Score: 0.21546911469919522
Test Accuracy: 0.9479809976435802
```

Our modified model has decreased the total number of parameters with compared to the general model. The reason for this with each convolutional layer the dimension of our image decrease. Which means by the time our image data reaches our fully connected layers it has much smaller dimensions. Which results in fewer dimensions. With this modification we require less computational power and improves our accuracy. The accuracy seems to be converging to a value of 0.9883 and validation accuracy is 0.9635. this modification significantly improved our accuracy.

But we can still see our model is overfitting.

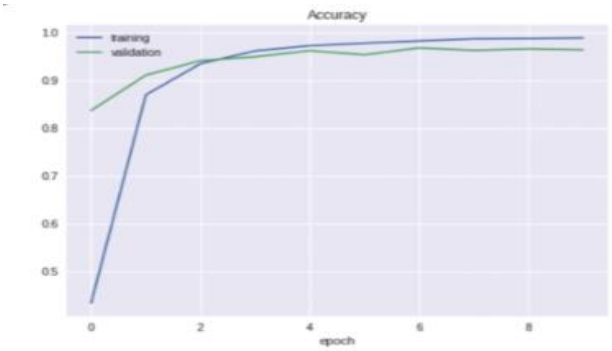


Fig.16 Accuracy vs Epoch in Extra Convolutional layer modified version

Adding Dropout Layer: As we can still see overfitting, adding a dropout layer can avoid it.

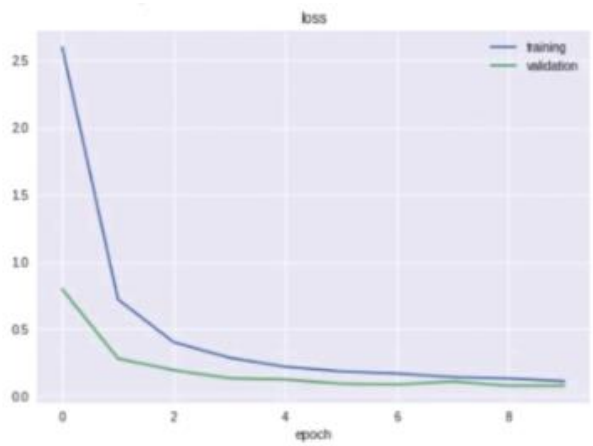


Fig.17 Loss vs Epoch in Extra Dropout layer modified version

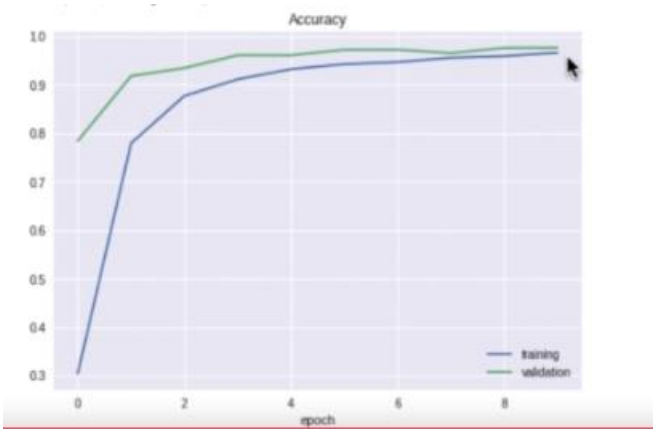


Fig.17 Accuracy vs Epoch in Extra Dropout layer modified version

```
Test Score: 0.13816216786407517
Test Accuracy: 0.9577988915753006
```

With all these modifications we can see our model improving slightly and preventing over fitting.

Data Augmentation: Is the process of creating new data to the model to use during our training. This is done by taking existing data and transferring, altering the image in various ways in a useful way.

The reason this augmentation data technique is useful as it allows our model to look at each image in our dataset from a variety of different perceptive. It allows to extract relevant feature more accurately and allows it to obtain more relevant feature image data. This method is more appropriate in our case because we have relatively less training images and more classes.

When we run this method on our dataset we get the following images

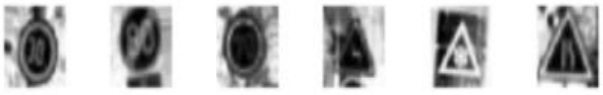


Fig.18 Dataset modified using Augmentation

As we can see augmentation generated images are in different variety wrt orientation.

Now when we finally run our model

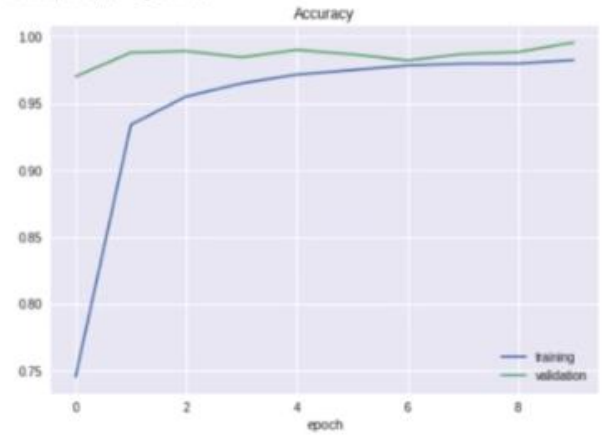


Fig.19 Accuracy vs Epoch in Augmented Dataset

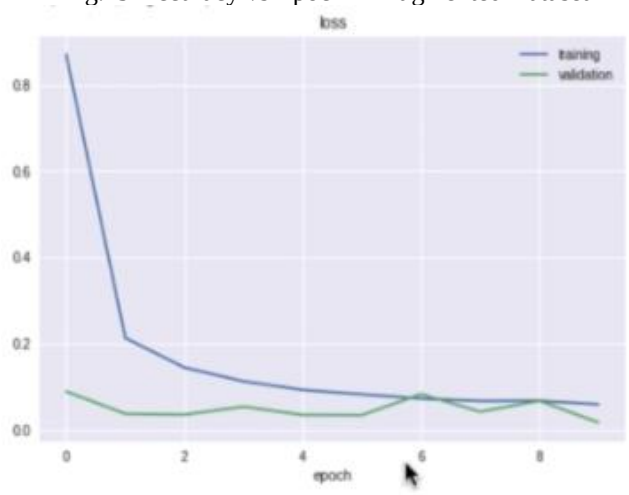


Fig.20 Loss vs Epoch in Augmented Dataset

```
Test Score: 0.10071537899478727
Test Accuracy: 0.9747426761584157
```

Fig.21 Test Accuracy of our model.

As we can see the final accuracy is about 97% and the gap between Validation accuracy and training accuracy is very small. In the same way the gap between Validation loss and training Loss is also small. Which means our model does a pretty good job in predicting the class value of given road sign image.

E. Testing our model

When we finally test our model with a random road sign taken from internet. Our model does pretty decent job by predicting that it belongs to its original class.

Fig.22 Test Image

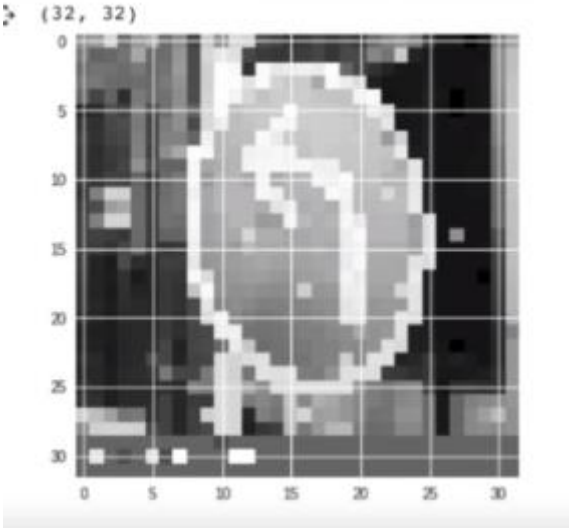


Fig.23 Applying Preprocessing techniques on test image.

So, our model predicts that this image belongs to class 34 which is 100 percent correct.

F. Results and Comparision

Model ID	Comparison with various classifiers		
	Classifier	***Features***	***Accuracy***
1	SVM (Cubic)	SURF	82.0%
2	SVM (Quadratic)	SURF	86.0%
3	ANN	SURF	40.0%
4	Decision Trees	SURF	66.0%
5	KNN	SURF	76.0%
6	Traditional CNN	SURF	93.0%
7	**CNN (proposed)**	**CNN Extracted**	**99.0%**

VI. CONCLUSION AND FUTURE WORK

When compared with other classifiers or traditional CNN our proposed classifier has better result. In this paper a detailed explanation of our CNN model is discussed which classifies the Road sign images. Our proposed methods and preprocessing techniques can be used as a role model if anyone wants to classify road sign images. As we all know this is a small part of Self driving car. In future we try to come up with techniques to optimize any feature implemented in self-driving car. if anyone uses our technique to predict road sign images, our model classifies with high accuracy and less computational time.

REFERENCES

[1] Zumra Malik and Imran Siddiqi, ”Detection and Recognition of Traffic Signs from Road Scene Image”,12th International Conference on Frontiers of Information Technology .

[2] C. G. Kiran, L. V. Prabhu, R. V. Abdu, and K. Rajeev, “Traffic sign detection and pattern recognition using support vector machine,” in Proc. ICAPR, 2009, pp. 87–90 .

[3] Broggi, A., Cerri, P., Medici, P., Porta, P. P., & Ghisio, G.,” Real time road signs recognition”, In IEEE Intelligent Vehicles Symposium, 2007, pp. 981-986.

[4] Soumenn Chakraborty and Kaushik Deb,”Bangladeshi Road Sign Detection Based on YCbCr color model and DtBs Vector”, 1st international Conference on Computer and Information Engineering, 26- 27 November, 2015.

[5] Maldonado-Bascon, S., Lafuente-Arroyo, S., Siegmann, P., GomezMoreno, H., & Acevedo-Rodriguez, F. J.,”Traffic sign recognition system for inventory purposes”, In Intelligent Vehicles Symposium, 2008, pp. 590-595.

[6] Huang H., Chen, C., Jia, Y. & Tang, S.,”Automatic detection and recognition of circular road sign”, In Proc. of International Conference on Mechtronic and Embedded Systems and Applications, 2008, pp. 626-630.

[7] De La Escalera, A., Moreno, L. E., Salichs, M. A., & Armingol, J. M., “Road traffic sign detection and classification”,IEEE Transactions on Industrial Electronics, 44(6), 848-859.

[8] Chen, S. Y., & Hsieh, J. W,”Boosted road sign detection and recognition”, In Proc. of Intl. Conference on Machine Learning and Cybernetics, 2008. pp. 3823–3826

[9] Jack Greenhalgh and Majid Mirmehdi,”Real-Time Detection and Recognitionof Road Traffic Signs”,IEEE Transactions on Intelligent Transport Systems, VOL. 13, NO. 4, December 2012.

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1-9, 2015.

[11] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: towards realtime object detection with region proposal networks,” Advances in Neural Information Processing Systems (NIPS), 2015.

[12] R. Girshick, “Fast r-cnn,” Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 1440-1448, 2015.