Traffic signs classification Project Report

Quanhan Sun[1], Guoyan LI[1], and Pengyu Huang[1]

Northeastern University

Abstract. In advanced driver assistance systems and autonomous driving, traffic signs recognition and classification played important roles [4]. Traffic signs provide valuable information about the road to help safe driving. This project used a dataset contains 43 classes with unbalanced class and develops conventional neural networks to solve this problem. There are two main parts in this project. The first one, data processing, including data augmentation and grayscale. The second one, classification. Using this model, it could tell the driver or the autonomous car the precise task with any other traffic marks.

Keywords: Deep Learning · Pattern Recognition · Convolutional Neural Network.

1 Background

A number of existing approaches to road-sign recognition have used computationally-expensive sliding window approaches that solve the detection and classification problems simultaneously or separately.

Classification has been approached by Neural Networks or Support Vector Machines [1]. Global sign shapes are first detected with various heuristics and color threshold, then the detected windows are classified using a different Multi-Layer neural net for each type of outer shape. These neural nets take 30x30 inputs and have at most 30, 15 and 10 hidden units for each of their 3 layers. While using a similar input size, the networks used in the present work have orders of magnitude more parameters. Also, a fast detector was used for round speed sign, based on simple cross-correlation technique that assumes radial symmetry. Unfortunately, such color or shape assumptions are not true in every country.

However, the existing approaches are expensive and time-consuming. In this project, we use conventional neural networks to perform detection and recognition simultaneously.

2 Introduction

In this project, we intend to achieve a multi-class classification problem.

The dataset we used is German Traffic Sign Recognition Benchmark [3], which was first used in a competition at IJCNN 2011 and it can be downloaded from public website. There are more than 50,000 images with 43 classes in this dataset and for each image data, the shape is 32*32*3. Firstly, we separated the whole dataset into a training dataset with 34799 examples and a validation dataset with 4410 examples. Secondly, we use a test dataset with 12630 examples to evaluate our creative model.

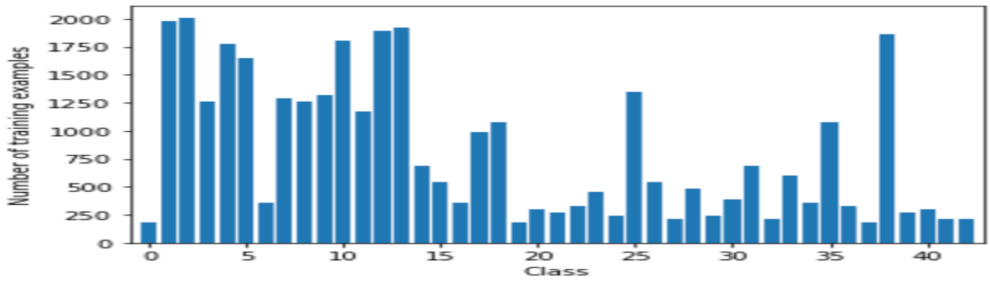

Fig. 1. The distribution of each class image

There are five main parts in this project: (1) Data pre-processing (2) Model building (3) Training process and validation process (4) Model comparison (5) Result.

3 Approach

3.1 Data Pre-processing

Grey-scale The usual preprocessing in this case would include scaling of pixel values to [0, 1] (as currently they are in [0, 255] range), representing labels in a one-hot encoding and shuffling. We will only use a single channel in our model, so we need to grayscale images instead of color ones. Figure2 illustrate what original and pre-processed images look like.

Fig. 2. Convert images into greyscale

3.2 Data Augmentation

The amount of data we have is not sufficient for a model to generalize well. It is also fairly unbalanced, and some classes are represented to significantly lower extent than the others. But we will fix this with data augmentation

Flipping First, we are going to apply a couple of tricks to extend our data by flipping. You might have noticed that some traffic signs are invariant to horizontal and/or vertical flipping, which basically means that we can flip an image and it should still be classified as belonging to the same class.

Fig. 3. Examples with horizontal or vertical flip

Some signs can be flipped either way — like Priority Road or No Entry signs.

Fig. 4. Examples with both horizontal and vertical flip

Other signs are 180 rotation invariant, and to rotate them 180 we will simply first flip them horizontally, and then vertically.

Fig. 5. Examples with 180 rotation

Finally, there are signs that can be flipped, and should then be classified as a sign of some other class. This is still useful, as we can use data of these classes to extend their counterparts.

Fig. 6. Examples can be flipped into another class

This simple trick lets us extend original 39,209 training examples to 59788. And it cost us nothing in terms of data collection or computational resources.

3.3 Model Architecture

In this project, we cited two model architectures. The first one is LeNet that was presented by Yann LeCun. Based on LeNet model, we changed slightly.

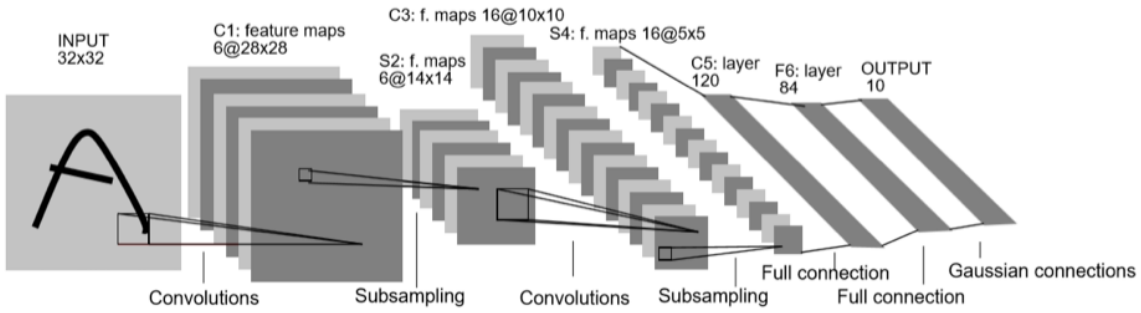

Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Fig. 7. The LeNet Model

In addition, we cited another model, which is aforementioned Pierre Sermanet / Yann LeCun paper [2]. It is fairly simple and has 4 layers: 3 convolutional layers for feature extraction and one fully connected layer as a classifier.

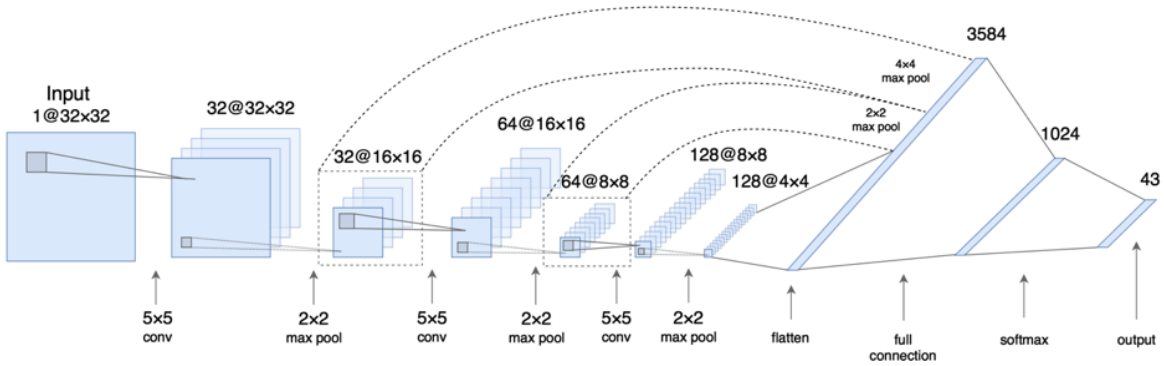

Fig. 8. The second model we use

As opposed to usual strict feed-forward CNNs.We use multi-scale features, which means that convolutional layers' output is not only forwarded into subsequent layer, but is also branched off and fed into classifier (e.g. fully connected layer). Please mind that these branched off layers undergo additional max-pooling, so that all convolutions are proportionally subsampled before going into classifier. Note that we collect all off convolutional layers' output, flatten and concatenate them before passing over to classifier.

Fig. 9. The Second Model Architecture

3.4 Regularization

We use the following regularization techniques to minimize overfitting to training data:

Dropout Dropout is amazing and will drastically improve generalization of your model. Normally you may only want to apply dropout to fully connected layers, as shared weights in convolutional layers are good regularizers themselves. However, we did notice a slight improvement in performance when using a bit of dropout on convolutional layers, thus left it in, but kept it at minimum:

Table 1. Dropout details of each layer

Case	Type	Size	Keep_p	Dropout
Layer1	$5\times5 Conv$	32	0.9	10%
Layer2	$5\times5 Conv$	64	0.8	20%
Layer3	$5\times5 Conv$	128	0.7	30%
Layer4	FC	1024	0.5	50%

L2 Regularization We ended up using lambda = 0.0001 which seemed to perform best. Important point here is that L2 loss should only include weights of the fully connected layers, and normally it does not include bias term. Intuition behind it being that bias term is not contributing to overfitting, as it is not adding any new degree of freedom to a model.

3.5 Validation and Comparison

We used our validation dataset,which includes 4410 images, to test our model and the result is shown below in Figure 10 for LeNet model and Figure 11 for 3-stage model.

we can find that accuracy of these two model do not have a significant difference, but according to the diagram above the 3-stage model has a better performance because it can avoid overfitting problem.

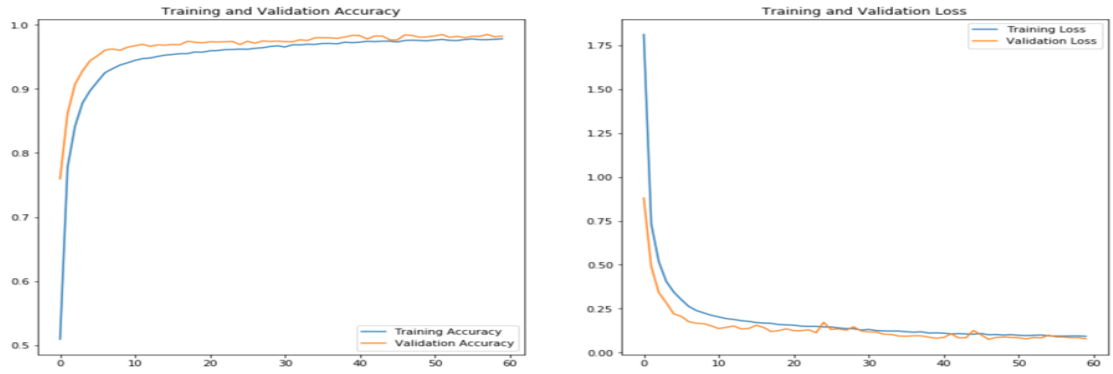

Fig. 10. The training and validation accuracy for LeNet Model

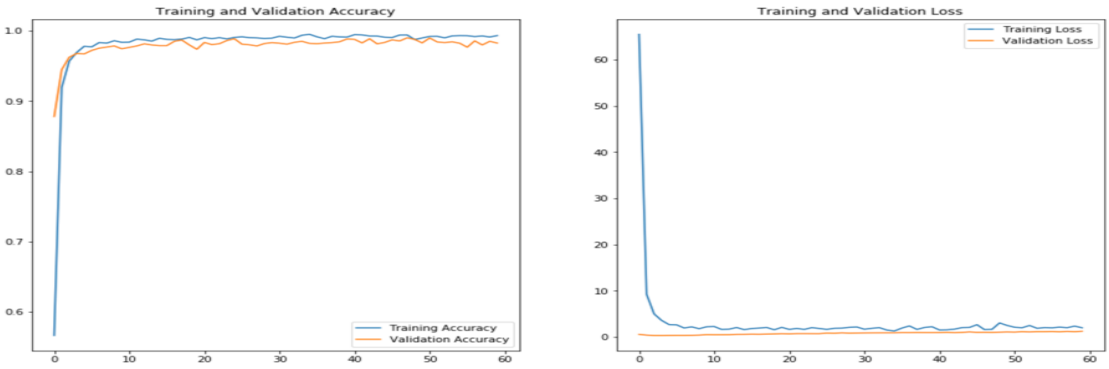

Fig. 11. The training and validation accuracy for 3-stage Model

3.6 Test and Result

we randomly choose 38 traffic sign pictures from the website. After labeling the test data with existing classes, we used our model to make an automatic classification. But as the figure 12 shown,we find that the speed limitation sign always have a relative lower accuracy than other traffic signs.

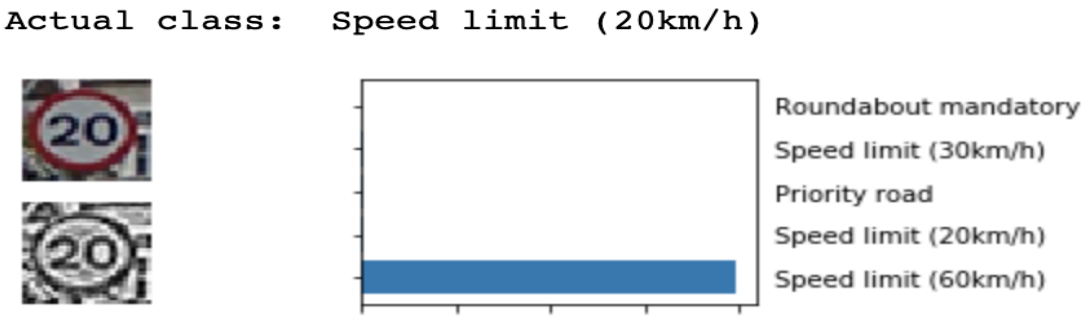

Fig. 12. The training and validation accuracy for LeNet Model

As shown in Figure1, the first index represents the sign of speed limitation. The total number of this kind of class is much less than others, which means we din not have enough speed limitation sign to train these two models. In case, we choose assign different weights to each classes, which can ensure our dataset can be balanced.

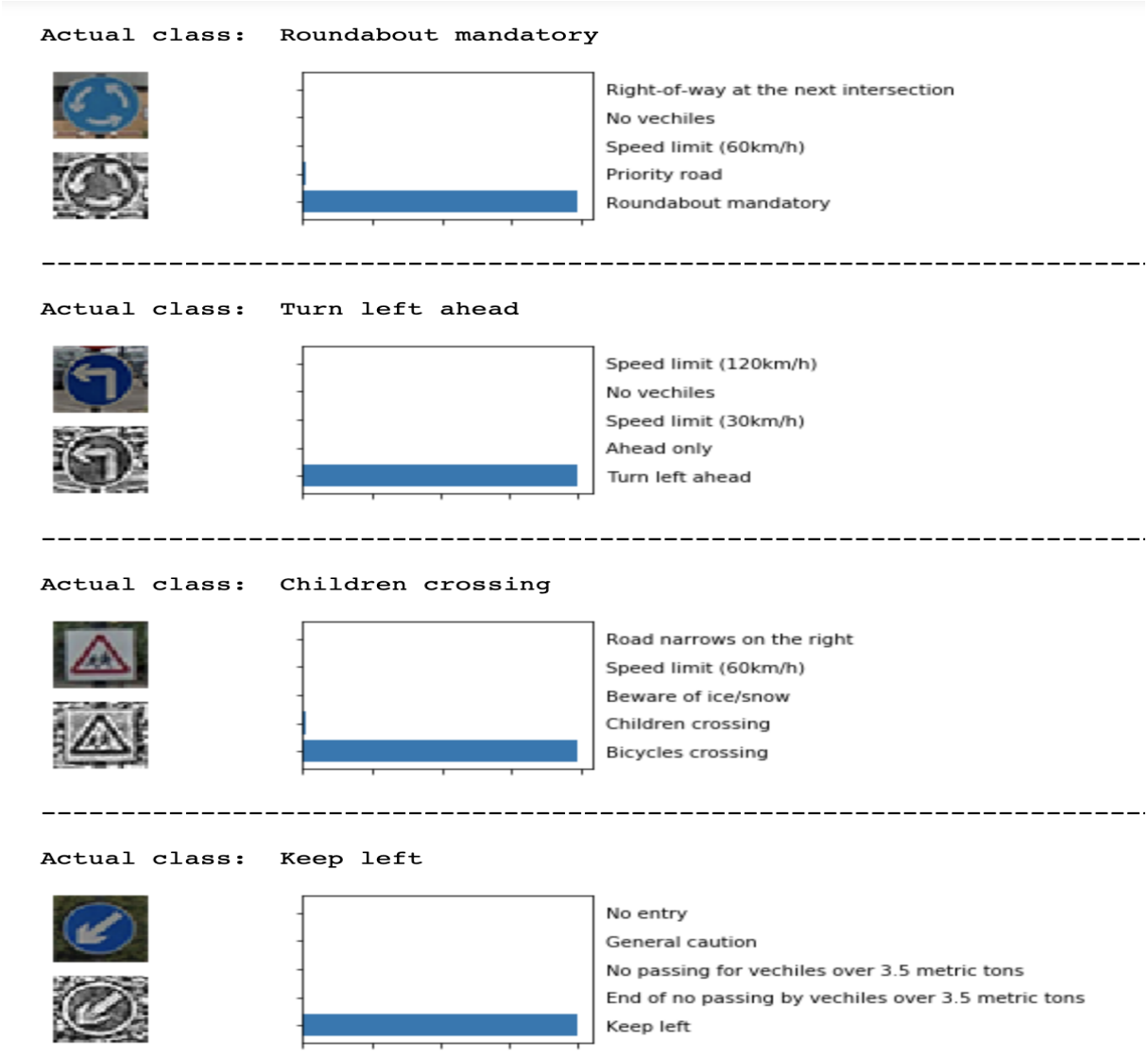

Fig. 13. The training and validation accuracy for LeNet Model

3.7 Conclusion

After a couple of fine-tuning training iterations this model scored 96.68% accuracy on the test set, which is not too bad. As there was a total of 12,630 images that we used for testing, apparently there are 85 examples that the model could not classify correctly.

Signs on most of the images either have artefacts like shadows or obstructing objects. There are, however, a couple of signs that were simply underrepresented in the training set — training solely on balanced datasets could potentially eliminate this issue, and using some sort of color information could definitely help as well.

References

1. Sergio Lafuente-Arroyo, Pedro Gil-Jimenez, R Maldonado-Bascon, Francisco López-Ferreras, and Saturnino Maldonado-Bascon. Traffic sign shape classification evaluation i: Svm using distance to borders. In *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, pages 557–562. IEEE, 2005.
2. Pierre Sermanet and Yann LeCun. Traffic sign recognition with multi-scale convolutional networks. In *IJCNN*, pages 2809–2813, 2011.
3. Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: A multi-class classification competition. In *IJCNN*, volume 6, page 7, 2011.
4. Jim Torresen, Jorgen W Bakke, and Lukas Sekanina. Efficient recognition of speed limit signs. In *Proceedings. The 7th International IEEE Conference on Intelligent Transportation Systems (IEEE Cat. No. 04TH8749)*, pages 652–656. IEEE, 2004.