# [EE240] Preliminary project report
# Simple Smart-Driving Skill Evaluation System

**Yuan-Pu Hsu**
#862057597

**Tianxiang Sun**
#862051319

## Abstract

In this project, we constructed multiple Convolutional Neural Network (CNN) models to recognize traffic signs and traffic lights efficiently for a driving skill evaluation application on mobile device. We proposed our own CNN architecture design, which was a 6-layer CNN model. The model were trained by German Traffic Sign Recognition Benchmark (GTSRB), and LISA Traffic Sign, and LISA Traffic Light dataset. The test accuracy reached 99.8% for recognizing traffic signs images taken in South California. Furthermore, we design our own iOS application, and completely integrate the CNN model both on the application and on the Google Compute Engine (GCE) to make it feasible for real-world real-time scenario successfully.

## 1 Introduction

Nowadays, over 90% of families in America own at least one vehicle. Driving has already been a big portion of American's daily life for decades. Therefore, we want to make a simple system to evaluate the driving skill of the driver. The user will get a score of their driving skill after finishing a tour, and the only thing needed is a smart phone. We believe it can help drivers to evaluate their driving skill and improve their skill to some extent. Furthermore, insurance company can also use the score to evaluate the credit of their clients, while those who has better driving skill should be benefited with lower insurance bill.
When it comes to details, firstly we build an application on our mobile device (in our case, iPhone6s) and use it as the platform. The user would be asked to put the phone behind the windscreen, and the appplication would automatically capture picture of front view to recognize different kinds of patterns on the road. In our project, the recognized pattern has been limited to traffic signs and traffic lights. In order to recognize patterns on the road, we first implement a mimic of VGGNet model, and found out the training time and running time would exceed one second which would not fit our proposed scenario. Therefore, we implement a 6-layer CNN model, which consisted of 6 convolutional layers, 3 max pooling layers and 3 dropout layers and a fully connected layer with softmax layer. We test our model after training them by using GTSRB and LISA dataset. From the evaluation, we found out that the testing accuracy of the 6-layer CNN model for GTSRB dataset was 96.4%, LISA traffic sign dataset is 99.8% and the LISA traffic light is 99.5%.

## 2 Related Work

Before we began our project, we had been inspired by the work of Shustanov, A. et al. [1] and Habibi Aghdam, H., et al [2]. In the work of the former one, the team described a revised end-to-end technology for detecting and recognizing traffic sign in real-time by CNN. It could use the speed received from the vehicle to predict not only the presence of the object but also the scale and its exact coordinates in the neighboring frame. In the latter work mentioned, the team proposed a light CNN which is designed for detecting traffic signs on high-resolution images. They trained the networks in two steps. First, the negative samples would be randomly selected from the training images. Second, it had been used to train the CNN using more appropriate data. The result of prediction has increased to 99.89% on GTSRB. Besides, we learned from the work of Simonyan, K.,et al.[3] and Kardkovacs, Z.T. [4] about how to increase the performance while meeting a large-scale image recognition setting. The team showed in the paper that they designed an architecture which increased the depth of the CNN model. Their architecture shows only 6.8% test error on top-5 testing.

## 3 Problem Formulation

The implementation of our project can be separate to three parts: iOS application (deploy on iPhone6s), Google Compute Engine with TensorFlow for speed-up development and accessing GPU computation, and the core CNN model for recognition. The system flow chart is shown in Figure 1.

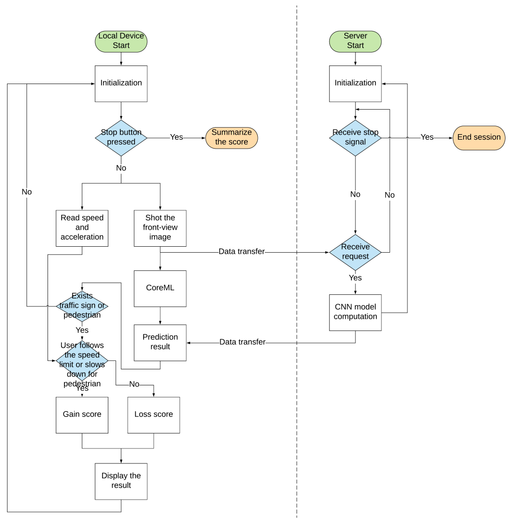


Figure 1: System design flowchart

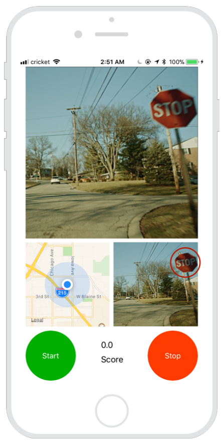


Figure 2: iOS app design

### 3.1 iOS Application

We first build a simple application for taking the front-view picture periodically, and record both the speed and acceleration at the same time via GPS and accelerometer. The picture will then pass through the trained CNN model installed in the bundle, or upload to the cloud service for the prediction. The result includes whether there existed a traffic sign or traffic light in the image, and what kind of sign it is. After the translation of the speed limit is done, the application can determine if the user was over speed or not. In the end, the application would give the user a high score if he/she follows the signs and vise versa. The illustration of the application design is shown in Figure 2.

### 3.2 Google Cloud Compute Engine

We build a virtual instance with a Nvidia P100 GPU on Google Compute Engine. In the instance, we install tools like Python and CUDA and TensorFlow etc., for fast implementation of training and tuning the CNN model. The application could communicate and upload image with PHP protocol.

### 3.3 Convolution Neural Network Model

#### 3.3.1 Preprocessing

Here we are using two kinds of datasets: GTSRB and LISA dataset. GTSRB contains 43 kinds of different traffic signs in Germany with 39,209 images in the set; LISA, on the other hand, contains 47 kinds of different US traffic signs with 7,855 annotations on 6,610 frames and 6 super-classes of traffic lights with total 43007 frames and 113888 annotations. Specifically, the images in LISA were

taken in South California, which is more close to our proposed scenario.
The training set would first be preprocessed with normalizing, centering and histogram equalizing to help accelerating the train process in the later section. In addition, they would also be augmented, by applying rotation, mirroring and flipping, etc., to mimic more possibilities in the real world cases. And then be separated into training set, validating set and testing set. The examples of the common type of sign images in the LISA dataset is shown in Figure 3.

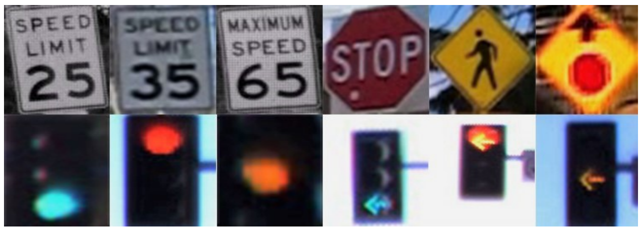


Figure 3: Images from LISA dataset

### 3.3.2 Model Architecture

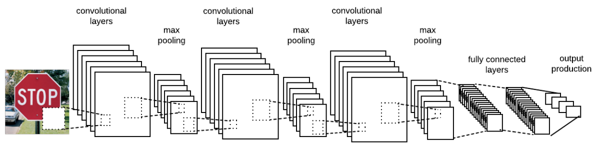
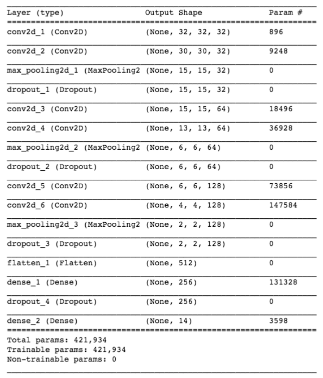


Figure 4: CNN architecture



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 30, 30, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 15, 15, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 15, 64)        18496
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 13, 13, 64)        36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 6, 6, 128)         73856
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 4, 4, 128)         147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 2, 128)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 2, 2, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               131328
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 14)                3598
=================================================================
Total params: 421,934
Trainable params: 421,934
Non-trainable params: 0
_________________________________________________________________
```

Figure 5: CNN layer detail with parameter

We constructed a 6-layer CNN. The model architecture is shown in Figure 4. Each convolutional block contains 2 convolutional layers with the same number of kernel and input/output nodes. Additionally, the each block would also followed by a max-pool layer and a drop-out layer. The features extracted by the previously mentioned layers would then be feed to a fully-connected layer with dropout. Finally, the softmax layer would aggregate the result and output the probabilities of the input image with regard to each class.

### 3.3.3 Model Training

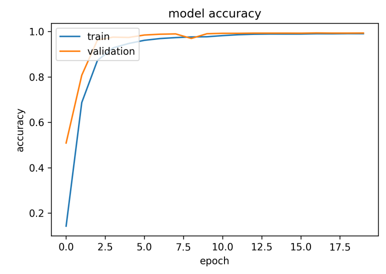


Figure 6: Training accuracy with GTSRB

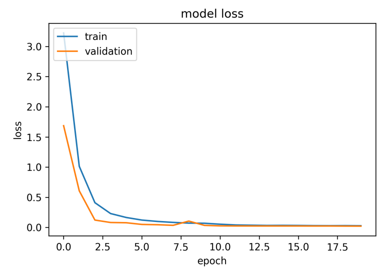


Figure 7: Training accuracy with GTSRB

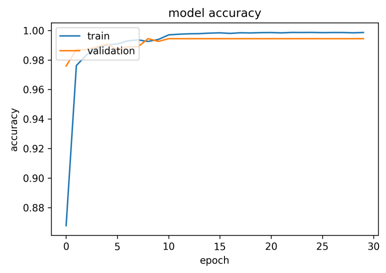


Figure 8: Training accuracy with LISA traffic sign dataset

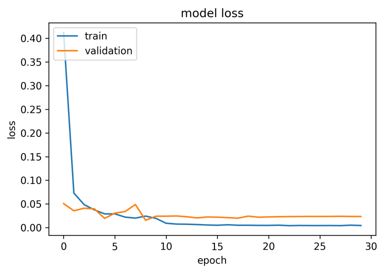


Figure 9: Training accuracy with LISA traffic sign dataset

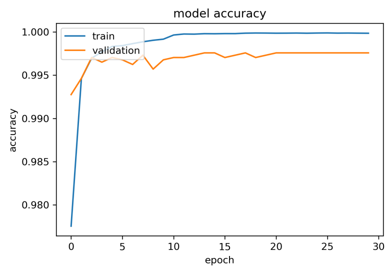


Figure 10: Training accuracy with LISA traffic light dataset

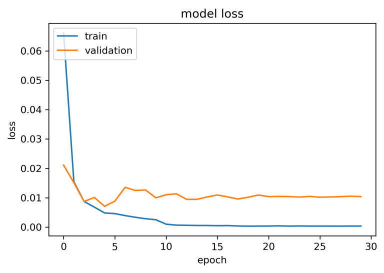


Figure 11: Training accuracy with LISA traffic light dataset

The models was trained by training dataset with augmented images, tuned with Stochastic Gradient Descent (SGD) and momentum to speed-up the training process. The training process took 2052 seconds per epoch with GTSRB dataset, 30 seconds per epoch with LISA traffic sign dataset, 201 seconds per epoch with LISA traffic light dataset. The training accuracy and loss by each epoch with three kinds of dataset was shown in Figure 6 to Figure 11. Among them, we can see the training accuracy could even reach 100%, and validation accuracy could be higher than 95%.

## 4 Evaluation

Table 1: CNN model test result

| Dataset | No. of classes | Test accuracy |
|---|---|---|
| GTSRB | 43 | 96.4% |
| LISA Traffic Sign | 14 | 99.8% |
| LISA Traffic Light | 6 | 99.5% |

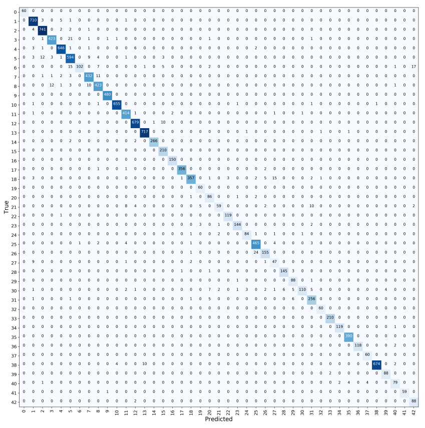


Figure 12: Testing confusion matrix with GTSRB

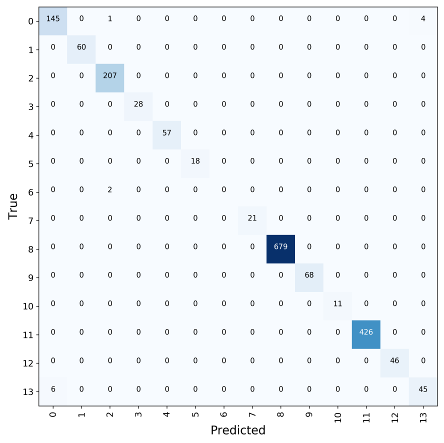


Figure 13: Testing confusion matrix with LISA traffic sign

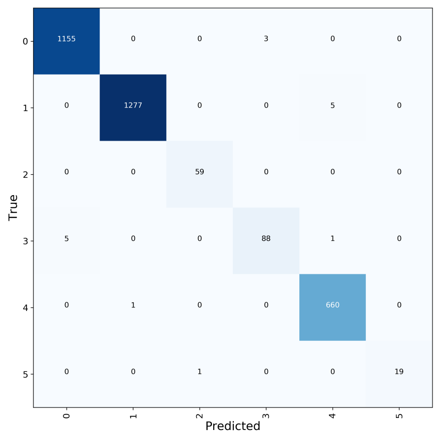


Figure 14: Testing confusion matrix with LISA traffic light

The test accuracy of our proposed model was shown in Table 1. The confusion matrix with regard to each datasets were shown in Figure 12 to Figure Figure 14 respectively. We can see that the proposed model would be able to reach at least 96.4% with GTSRB dataset. Moreover, for our proposed scenario, we focused more on the LISA dataset, and it could even reach higher accuracy, 99.5% and 99.8% for LISA traffic light and LISA traffic sign dataset. On the other hand, the model running time on the transformed coreML model was only 70 ms in average. The test result can support the feasibility of the real-time scenario that we proposed.

## 5 Contribution

The contributions of our projects can be summarized as follows:

- We proposed our own CNN architecture which was a 6-layers CNN with 3 max-pooling layers, 3 drop-out layers and 1 fully-connected layer with softmax layer to conclude the results. The model was trained by GTSRB, and LISA dataset for different case of scenarios.
- We evaluated our models to see the performance. We showed that our models can reached at least 96.4% test accuracy with GTSRB dataset, 99.5% with LISA traffic light dataset and 99.8% with LISA traffic sign dataset.
- The models developed with the environments of Keras and TensorFlow are both deployed and tested on local PC, Google Compute Engine, and the mobile device.
- We designed our own iOS application by Swift deploying on iOS 11.2 served as the platform. In addition, we're able to measure the speed and acceleration, and compare with the recognized traffic sign from road image in real-time.

## 6 Acknowledgements

In our original implementation, we mimic the VGGNet architecture based on a GitHub repository (`https://github.com/vamsiramakrishnan/TrafficSignRecognition`). The model achieved high accuracy with GTSRB dataset, but it'll not fit our real-time scenario. Therefore, we reference the CIFAR-10 model in the tutorial of TensorFlow official website (`https://github.com/vamsiramakrishnan/TrafficSignRecognition`), which was a 2-layer CNN model. We extend the complexity of each convolutional layers and conclude the result by one fully-connected layer with softmax layer. Finally, the training and testing result and running speed both high enough for our proposed system.

## References

[1] Shustanov, A., Yakimov, P., (2017). CNN Design For Real-Time Traffic Sign Recognition. *Procedia Engineering.* Vol.201,pp.718-725.

[2] Habibi Aghdam,H., Jahani Heravi,E., Puig,D. (2016) A practical Approach for Detection and Classification of Traffic signs using Convolutional Neural Networks. *Robotics and Autonomous System*, Vol. 84, pp. 97-112.

[3] Simonyan, K., Zisserman, A., (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *ARXIV.* eprint arXiv:1409.1556

[4] Kardkovacs, Z.T., Paroczi, Z.,Siegler,A.,(2011) Real-Time Traffic Sign Recognition System. *2nd International Conference on Cognitive Infocommunications.*