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

Yuan-Pu Hsu
#862057597

Tianxiang Sun
#862051319

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 the details, we want to build an app in the smart phone (in our case, iPhone). The user can place the device behind the front windscreen, and let it automatically analyze the image and recognize different patterns on the road. In this project, we restrict the targets within the vision to traffic lights, stop signs, speed limit signs, and pedestrians. Besides, we will take advantage of the GPS and accelerometer on the phone to calculate the speed and acceleration of the car. After comparing the speed of the car and patterns gotten by the mobile device, we can evaluate his/her driving score by the information. For example, if the user did not slow down while meeting a pedestrian, he/she would lose some points; if the user drive smoothly for a period of time, he/she would gain some points.

There are several researches regarding to recognition of traffic signs as listed in the reference. Mainly, we are going to implement Convolutional Neural Network (CNN) for pattern recognition with the method by Shustanov, A. et al. [4], Habibi Aghdam, H,. et al. [2], and optimize the speed for real-time scenario with the method of Kardkovacs, Z. et al. [3] We want to use the dataset called German Traffic sign benchmark as the training set of traffic signs. On the other hand, we are using cascade classifier by Angelova, A. et al. [1] to detect pedestrians for low latency with Caltech dataset. We will utilize the common tools for implementing computer vision and machine learning like TensorFlow, and AWS to build the proposed system. Meantime, we would also compare the efficiency of computation under different real-time scenario, such as whether compute locally with GPU on the phone, or upload to the cloud via LTE.

2 Implementation

The implementation of our project can be separate to three parts: iOS application (deploy on iPhone6s), Amazon Web Services (AWS) 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.

2.1 iOS Application

We first build a simple app for taking the front-view picture periodically, and record both the speed and acceleration at the same time via MapKit. 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

Preprint. Work in progress.

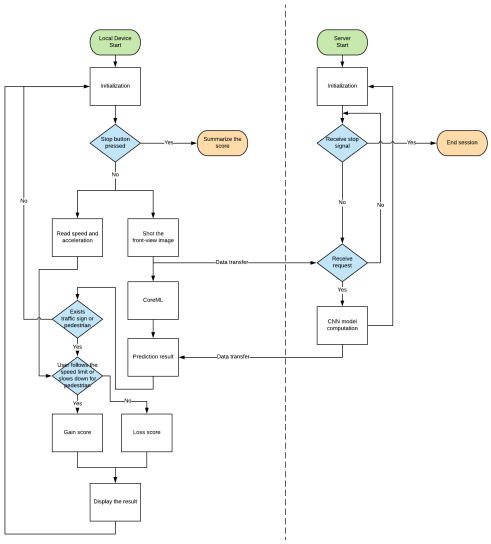

Figure 1: System design flowchart

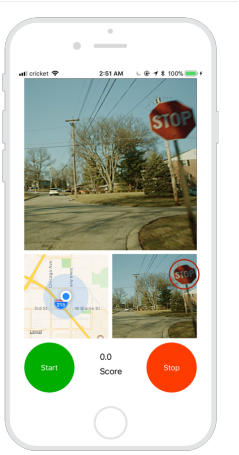

Figure 2: iOS app design

whether there existed a traffic sign or pedestrian in the picture, and what kind of sign it is. After the translation of the speed limit is done, the app can determine if the user was over speed. In the end, the app would give the user a high score if he/she follows the signs and slow down for the pedestrian and vise versa.

2.2 AWS Server with TensorFlow

We build a Ubuntu virtual machine on the AWS server with both CPU and GPU deployment. In the machine, we installed tools like Python, TensorFlow and CUDA, etc., for fast implementation of train and tune the CNN model. After the model was trained on the server, we On the other hand, the server would allow the local device to upload the input image for prediction and then pass back the result of prediction after the computation.

2.3 Convolution Neural Network Model

2.3.1 Preprocessing

Here we are using German Traffic Sign Benchmark (GTSB) as the training set, which contains 43 kinds of different traffic signs with around 34000 images in the set. The training set would first be processed with mean, centering, normalizing, and histogram equalizing to help accelerating the train process in the later section. In addition, for augmenting the dataset, we also applied rotating and mirroring to mimic more possibilities in the real world cases. Then separate into 3 sets: training set, validating set and testing set in order to train the model. The distribution of each class of data is shown in Figure 3, and three kinds of the common type of sign images in the dataset is shown in Figure 4.

2.3.2 Model Architecture

We would construct our own CNN model based on VGG Net [5], with a linear classifier. Specifically, we adopted VGG-5 architecture with four VGG convolution layers, a multi scale concatenate layer, two fully connected layers and a logits layer. With the multi scale architecture that subsampling the output of each VGG block and concatenate before fully connected layer, the lower order feature in the image would still be possible to vote for the final result. While the traffic-sign feature in our scenario might serve as a small part in the whole image, this way should be a better approach to treat those and have more accurate results. Besides, we also implement Spatial Transformers for achieve invariance in scale, and the Softmax activation function for classification.

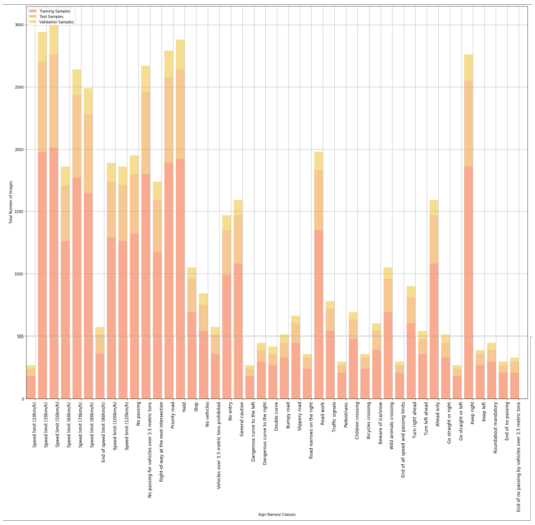

Figure 3: Data distribution

Figure 4: Images of the traffic signs

2.3.3 Model Training

In order to train the CNN model more efficiently, we use the AWS EC2 server to conduct all the complex computation. We would tune the learning rate, number of iterations, batch size, number of epochs, dropout rate, and regularization factor, etc., to construct a reliable model structure.

3 Proceeding works

After completing the training process, we would conduct the model into experiment. Our model would be translate into coreML model for implementation on the iOS device. To test computation speed and accuracy both on the cloud service and local device. On the other hand, we would also work on speed-up and optimize the model for securing real-time scenario.

References

[1] Angelova,A., Krizhevsky, A., Vanhoucke, V. Real-Time Pedestrian Detection With Deep Network Cascades. *Google research.*

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

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

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