Behavioral Cloning for Self Driving Car

1 Introduction

The goal of the project is to auto control the steering of a moving car by predicting steer angle through deep neural network. In order to achieve this, the user first drives the car on the tracks and captures train images and other control parameters like steer angle. These obtained training vectors are fed to CNN as <train image, steer_angle> tuples to train the network. Once the network is trained, the user drives the car in autonomous mode and based on the image sampled at the moment, the trained CNN predicts the steer angle which is used in controlling the car. The project is implemented using Keras API with Tensorflow backend for the implementation of different layers of the network. Below sections describe the architecture used for the CNN , the data collection methodology, model training and results.

2 CNN Architecture

The model uses convolutional neural network that comprises of 6 layers of convolutional layers followed by fully connected layers of neural network . Below digram shows different layers used in the model with details on number of features sets(depth), kernel size, activation function used. The model architecture used in the project is similar to the CNN architecture discussed in [1] with some modifications as addition of some layers

Output 1
FC 10, relu activation
FC 50 , relu activation
FC 100, 0.5 Dropout , relu activation
Flatten 2880 with 0.5 Dropout
Conv features 64@3x15, 3x3 filter, RELU activation
Conv features 64@5x33, 3x3 filter, RELU activation
Conv features 64@7x35, 3x3 filter, RELU activation
Conv features 48@9x37, 5x5 filter, RELU activation
Conv features 36@21x77, 5x5 filter, RELU activation
Conv features 24@46x158, 5x5 filter, RELU activation
Cropping 3@96x320
Normalize with zero mean

and scaled in range [-1, 1]
Input YUV 3@160x320

3 Data Collection

The data is collected by driving the car in manual model on Track1 and Track2. For the project, the training data for track1 is used as provided, further one lap of track2 is recorded and mixed with track1 data. Driving on the track2 is very difficult in the simulator, so the collected data(for track2) is not very good in the sense, they are not center road driven or within lane for track2. Below shows the center image and augmented left, right and left shifted images used in the training.

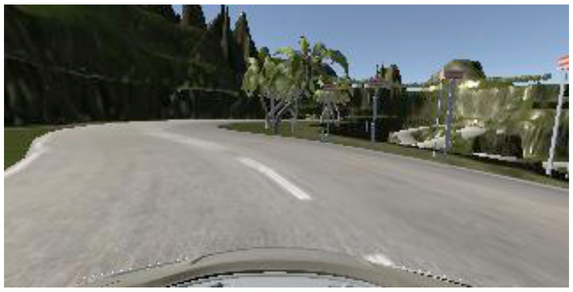
Illustration 1: center image as seen from center camera

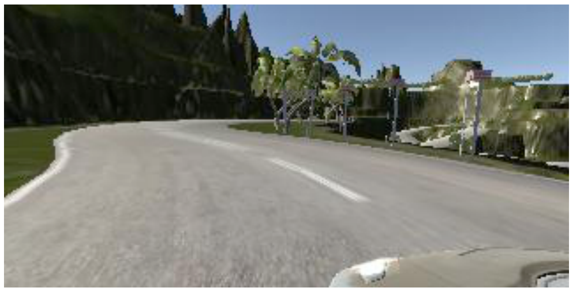
Illustration 2: Left view image as seen though left camera sensor

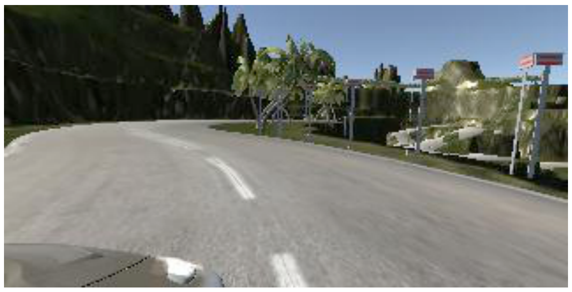
Illustration 3: Right view image as seen from right camera

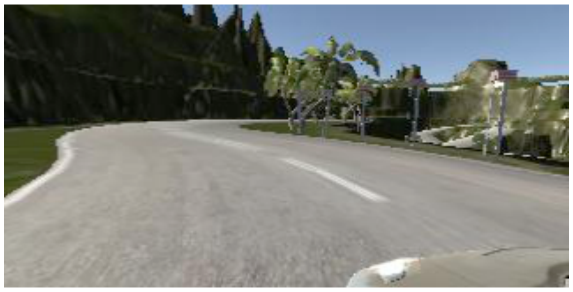
Illustration 4: left shifted by 16 pixels from left view

4 Data Augmentation and Model training

Since the data collected are not sufficient for the network training, each batch of data are augmented using left/right camera views' images and lateral horizontal shifted images. These images act as recovery samples through which the car learns to correct itself if it deviates or makes mistakes along the route. The lateral horizontal shift was drawn using random normal distribution with zero mean and 4 pixels std deviation. The steer angle for these augmented images are adjusted from the central image's steer angle as appropriate.

Adam optimizer is used along Mean Squared Error(MSE) cost function for the training of the model. L2 kernel regularizer with dropout(in some of FC layers) are used to prevent the overfitting of the network. Python generator is used to generate each batch of training and cross validation images on the fly which helps to save memory for the storage of data. 60 epochs are used to train the model and the model is saved as keras h5 file at the end of training. The number of epochs are selected based on the experimentation as when the model stops learning or learns very slowly, indicating the convergence of the model to its optima. In the SGD, the model never converges to optima rather it oscillates around optimum region. Below plot shows training and validation errors obtained from the history of the model once the model is trained for 60 epochs.

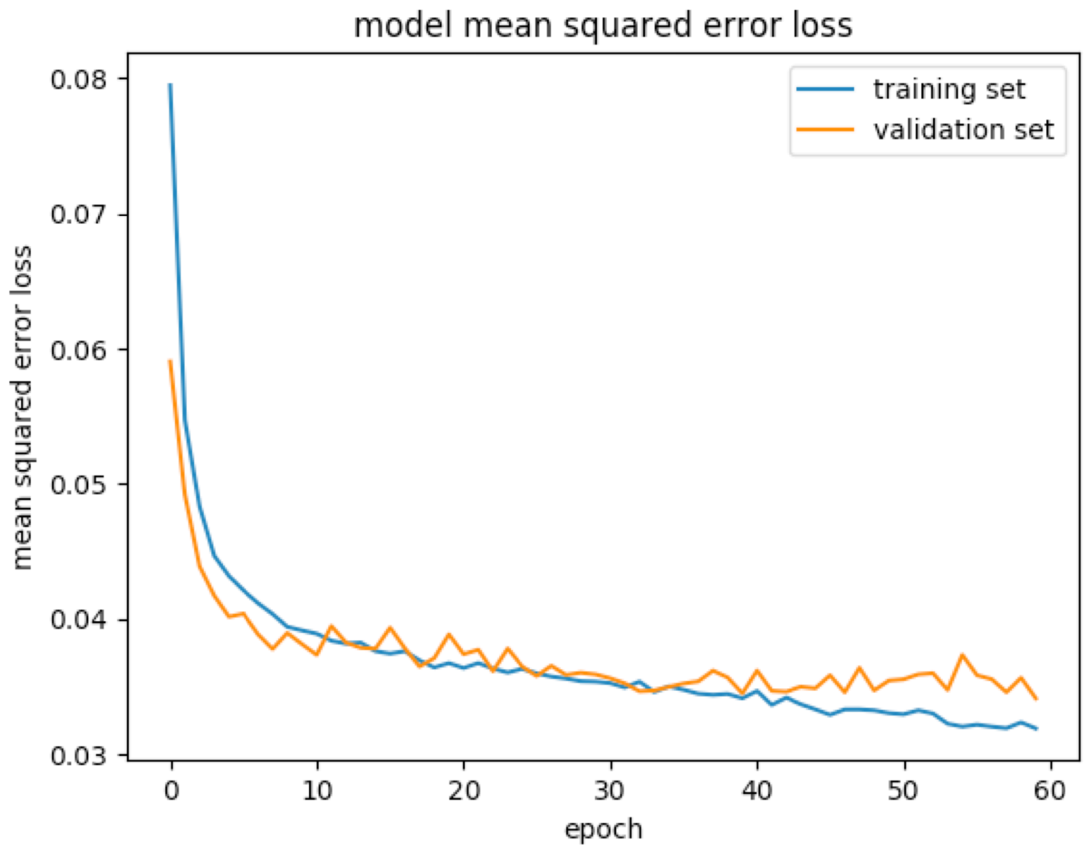

Illustration 5: Error curve showing train and cross validation error

General steps for the training process of the model is as described below, though some of the steps like gradient calculation, and weights' updates are done using Keras API:

```
train, val = split with 80% as train samples
for each epochs:
    shuffle the train data
    for each batch:
        for each image:
            Forward pass through the network, followed by error BackProp
            Accumulate the weights' gradient
        update the weights using corresponding accumulated gradients
        Evaluate the model using Validation data
```

During Autonomous driving mode, the saved model file(.h5) is loaded and used for the prediction of steer angle from the center images. These predicted steer angles are used to auto steer the car.

5 Result Discussion

The car drives very well in autonomous mode for track1. Auto mode is very smooth on the track1 without any issue and most of the time it positions itself at the center of the road, since the training data used for track1 is very well behaved. Driving on track2 is very difficult and it requires a lot of effort to gather well behaved(centered) data. Track2 training data collection show rash driving behavior which is also manifested in autonomous mode on track2.

Below files are submitted along with the project:

1. model.py: contains code for model training, batch data generator with augmentation

   ```
   python model.py <input image directory> <model file to be saved as>
   ```

2. drive.py: Modified file for driving on the track in Autonomous mode.
3. model.h5: saved trained model file to be used along with drive.py

   ```
   python drive.py model.h5
   ```

4. video.mp4: recorded video while driving on Track1 in autonomous mode @speed 30mph

   video_track2.mp4: recorded video while driving on Track2 in autonomous mode @20mph

6 Conclusion

Multi layer deep neural network can successfully be used to control parameters like steer angle, speed etc. in self driving car. In order to achieve good performance and well behaved self driving, training data collection, representing wide variety of driving conditions, plays a very crucial role apart from powerful model architecture design. The autonomous driving on the track1 is very smooth due to the quality of data used, whereas bad driving behavior during training sample collection on track2 resulted in very sharp turn and rash driving on track2. Further improvement on track2 can be made by collecting centered driving and smooth turn data along the curve. Proper value of steer angle correction for the augmented images also improves the performance, so proper method needs to be investigated to be more precise in angle corrections.

7 References

1. End to End Learning for Self-Driving Cars
Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, Karol Zieba