Vehicle Auto Steering Using Deep Learning

Shrikant Chaudhari
Manish Kondala

Outline

- Introduction
- Softwares Used
- Approach
- Results and Evaluation
- References

Introduction

- Concepts of deep learning and convolutional neural networks are applied to teach the computer to drive car autonomously
- CNN methodology has been implemented for the estimation of steering angle
- Images from the front mounted car cameras (input to CNN) are fed into a CNN which then computes a proposed steering command
- Once trained, the network can generate steering angles from the video images of a single center camera

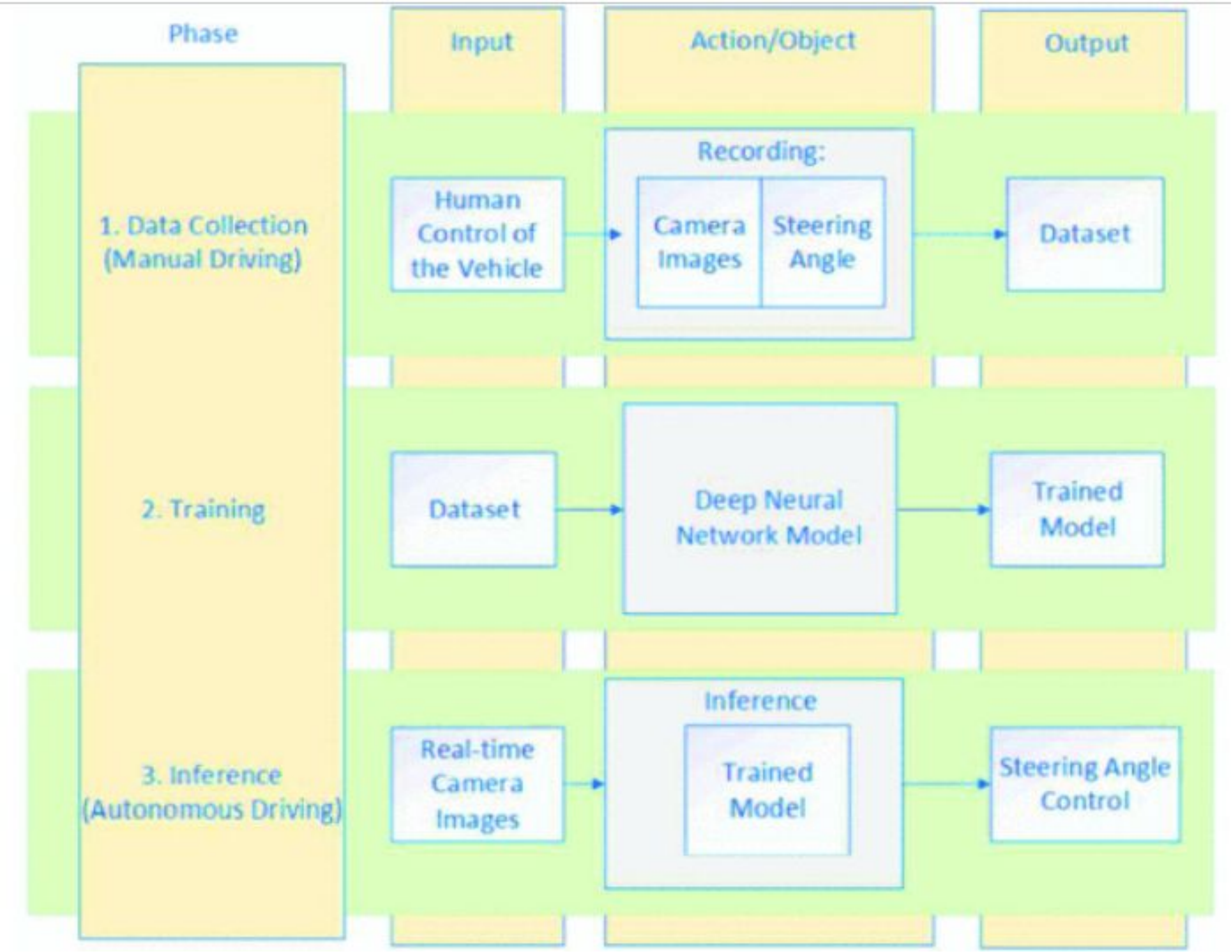
Softwares Used

- The data collection is performed using Udacity self-driving car simulator designed by Unity (game engine)
- VSCODE and Google Colab for scripting and model training
- Python Socketio and flask libs to connect the trained model to the simulator

Approach

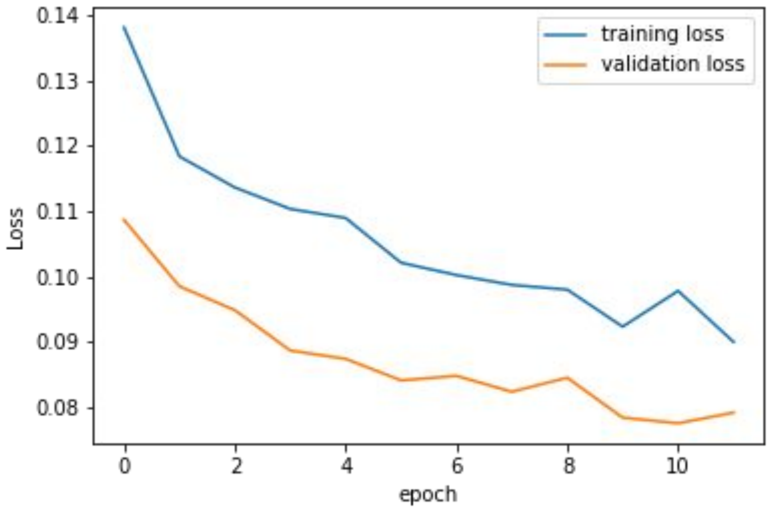
- Udacity open source car simulator is used for data generation. It provides two options: training mode and autonomous mode
- For preparing the deep neural network, pictures are procured from all of the three cameras - centre, left, and right
- Image augmentation is done on the raw data set, to increase the number of images and for the model to easily adapt to different types of images to learn
- Zoomed, Panned, brightness altered, mirrored images are generated from the original images using batch generator
- The deep neural network for independent driving is prepared on this dataset and can anticipate the directing point
- At last, this prepared model is utilized for inference, that is a continuous execution of the independent vehicle in a similar stimulator environment

Approach


Phase
Input
Action/Object
Output
1. Data Collection (Manual Driving)
Human Control of the Vehicle
Recording:
Camera Images
Steering Angle
Dataset
2. Training
Dataset
Deep Neural Network Model
Trained Model
3. Inference (Autonomous Driving)
Real-time Camera Images
Inference
Trained Model
Steering Angle Control

Results and Evaluation

- Training and validation loss during model training phase was 0.098 and 0.0716 respectively
- With less training data, better results are obtained when car is run on first track (other than unseen track)
- To overcome overfitting problem, drop out technique is implemented

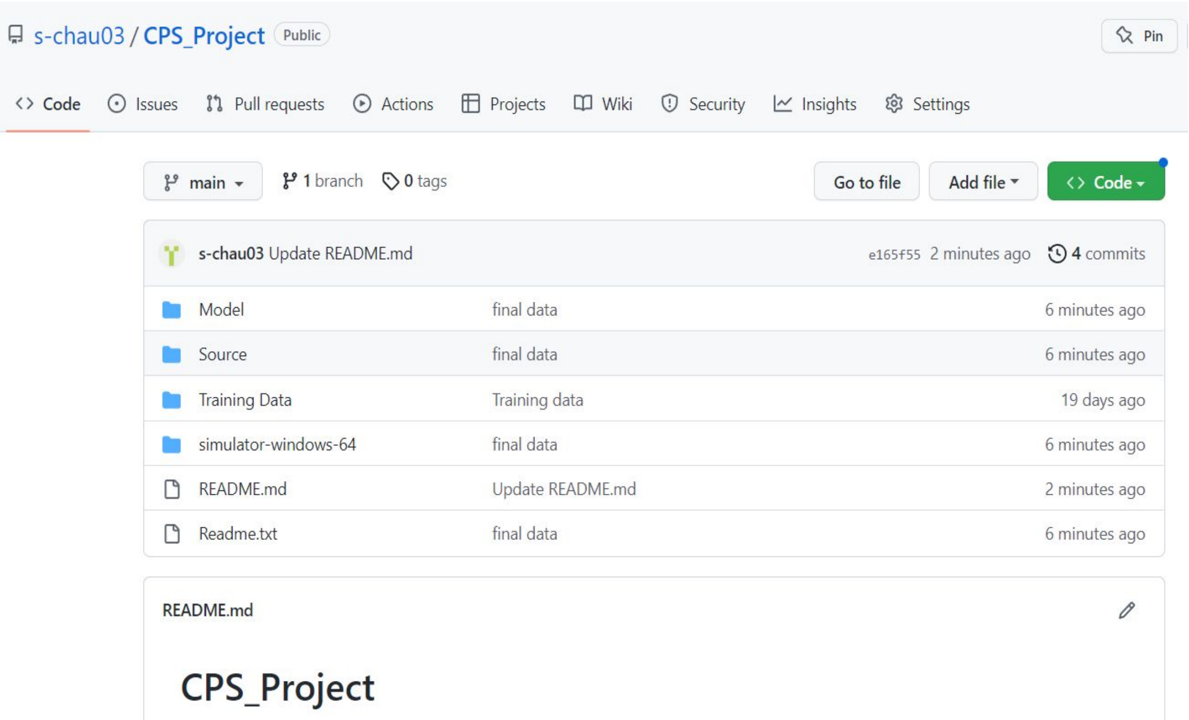


Results and Evaluation

```
<ipython-input-33-115b913bba99>:1: UserWarning: `Model.fit_generator` is deprecated and will be r
  h = model.fit_generator(batch_generator(X_train, y_train, 100, 1),
Epoch 1/12
100/100 [==============================] - 189s 2s/step - loss: 0.1387 - val_loss: 0.1124
Epoch 2/12
100/100 [==============================] - 187s 2s/step - loss: 0.1302 - val_loss: 0.0955
Epoch 3/12
100/100 [==============================] - 202s 2s/step - loss: 0.1168 - val_loss: 0.0882
Epoch 4/12
100/100 [==============================] - 184s 2s/step - loss: 0.1102 - val_loss: 0.0906
Epoch 5/12
100/100 [==============================] - 182s 2s/step - loss: 0.1065 - val_loss: 0.0794
Epoch 6/12
100/100 [==============================] - 183s 2s/step - loss: 0.1060 - val_loss: 0.0783
Epoch 7/12
100/100 [==============================] - 183s 2s/step - loss: 0.1000 - val_loss: 0.0775
Epoch 8/12
100/100 [==============================] - 183s 2s/step - loss: 0.0998 - val_loss: 0.0797
Epoch 9/12
100/100 [==============================] - 183s 2s/step - loss: 0.0982 - val_loss: 0.0786
Epoch 10/12
100/100 [==============================] - 181s 2s/step - loss: 0.0964 - val_loss: 0.0729
Epoch 11/12
100/100 [==============================] - 182s 2s/step - loss: 0.0975 - val_loss: 0.0757
Epoch 12/12
100/100 [==============================] - 181s 2s/step - loss: 0.0898 - val_loss: 0.0716
```

Github Code Repository

- https://github.com/s-chau03/CPS_Project



References

- Singh, S. (2015). *Critical reasons for crashes investigated in the national motor vehicle crash causation survey* (No. DOT HS 812 115).
- Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., ... & Zieba, K. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*..
- Martin Buehler, Karl Iagnemma, and Sanjiv Singh. The DARPA urban challenge: Autonomous Vehicles in City Traffic, volume 56. Springer, 2009. Springer Tracts in Advanced Robotics
- Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. End To-end learning of driving models from large-scale video datasets. In IEEE Int. Conf. Comput. Vis. Pattern Recog. (CVPR), pages 3530–3538, July 2017.
- Isele, D., Rahimi, R., Cosgun, A., Subramanian, K., & Fujimura, K. (2018, May). Navigating occluded intersections with autonomous vehicles using deep reinforcement learning. In 2018 IEEE International Conference on Robotics and Automation (ICRA) (pp. 2034-2039). IEEE.
- Maqueda, A. I., Loquercio, A., Gallego, G., García, N., & Scaramuzza, D. (2018). Event-based vision meets deep learning on steering prediction for self-driving cars. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5419-5427).
- Kumar, N., Chilamkurti, N., & Park, J. H. (2013). ALCA: agent learning–based clustering algorithm in vehicular ad hoc networks. *Personal and ubiquitous computing*, *17*(8), 1683-1692.
- Lee, M., & Atkison, T. (2021). Vanet applications: Past, present, and future. *Vehicular Communications*, *28*, 100310.