Vehicle Auto Steering Using Deep Learning

Shrikant Chaudhari
Department of Electrical and Computer Engineering
Stony Brook University
shrikant.chaudhari@stonybrook.edu

Manish Kondala
Department of Electrical and Computer Engineering
Stony Brook University
kondala.manish@stonybrook.edu

ABSTRACT

Vehicular technology has lately gained increasing popularity, and independent driving is a hot topic. To achieve safe and dependable intelligent transportation systems, accurate positioning technologies need to be constructed to factor in the different types of misgivings similar as random pedestrian, arbitrary objects, and types of roads and their settings. For a car to become completely independent, the upcoming technologies need to be accurate enough to gain public trust and show immense delicacy in their approach to working issues. Here the concepts of deep learning and convolutional neural networks are applied to teach the computer to drive car autonomously. In this project we used the image augmentation technique to expand our data set by implementing various factors.. To develop our behavioral cloning model, we have used the Nvidia Convolution Neural Network. Once trained, the network can generate steering angles from the video images of a single center camera. The data collection is performed using Udacity self-driving car simulator designed by Unity.

Keywords

Convolution Neural Network; Image Augmentation; Behavioural Cloning; Deep Learning;

1. INTRODUCTION

Recent advances in autonomous driving have been fueled by modern deep learning methods, whereby driving controllers are typically trained on extensive datasets of real-world recordings and simulated environments. The availability of these datasets together with advances in deep learning has enabled improvements in computer vision technologies that are essential to the success of autonomous driving, such as semantic segmentation, object detection, tracking and motion estimation. Self-driving vehicles should work under an extensive variety of lighting conditions, and in this manner it is significant that utilized vision sensors offer high unique reach and high responsiveness, empowering short openness times to limit movement obscure. Occasion cameras, for example, Dynamic Vision Sensors can offer benefits under conditions that are challenging for traditional cameras. As opposed to customary examined, outline based cameras, occasion cameras produce a surge of nonconcurrent timestamped address occasions that are set off by neighborhood splendor changes at individual pixels.

Currently, there are more than 1,400 autonomous cars, trucks, and other vehicles in the testing phase initiated by more than 80 companies across 36 states in the United States of America. California is listed among those states that have deployed AVs on public roads. AV control has made some advancement in recent years, and many AV vendors have promised commercial production on a large scale within a period of two to three years. AVs currently have a massive impact on the automotive industry [1]. In AV steering control, lateral and longitudinal motions constitute the major components of vehicle motion control. Steering of the vehicle, control of the lateral motion of AVs that aim at controlling the position of the vehicle in the lane, and other lateral actions like changing of pathway and avoidance of collision while manipulating pedals of the vehicle are aspects of longitudinal motion.

Deep Learning has made a positive impact on the control of AVs, particularly in terms of the steering angle prediction due to its ability to effectively process unlabeled raw data. Deep Learning understands the world through analyzing the context of a scene, while focusing on essential objects and observing them at hierarchical levels – from small objects with higher resolution to large objects with lower resolution. Therefore, when analyzing a scene, DL is reasonably insensitive to variations of environmental conditions, yet requires a large amount of high-quality data to achieve high accuracy. Neural networks can learn complex interactions between features, which is beneficial for autonomous driving in dynamic environments. Steering a car through traffic constitutes a complex task that is very hard to cast into algorithms. Thus, researchers turn to train Artificial Neural Networks with a stream of data generated by front-facing cameras yielding associated steering angles.

This paper portrays a CNN that goes on the most distant side of pattern acknowledgment. It learns the total handling pipeline needed to guide the vehicle. The CNN is utilized to map raw pixels from three cameras in a udacity stimulator and information is stored as csv documents. These information documents are utilized to prepare the neural network and can foresee steering angles. In this manner another model design was taken, and the picture expansion and processing part was improved subsequently making the vehicle to run on the two tracks in the Autonomous Mode.

2. RELATED WORK

Developing robust policies for autonomous driving is a challenging research problem. Highly engineered, modular systems demonstrated incredible performance in both urban and off-road scenarios [3]. Another approach to the problem is to directly map visual observations to control actions, tightly coupling the perception and control parts of the problem. Xu et al. [4] proposed to leverage large-scale driving video datasets and to do transfer learning to generate more robust policies. The model showed good performance but was limited to only a set of discrete actions and was susceptible to failures in undemonstrated regions of the policy space.

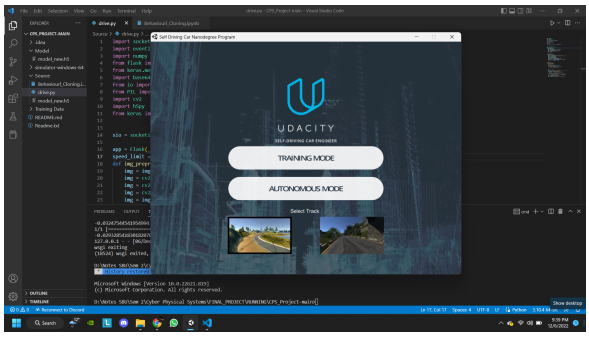
The authors in [5] have worked on the effectiveness of Deep Reinforcement Learning to handle intersection problems.They have worked on systems abilities to make the system learn active sensing behaviors to enable navigating safely in the case of occlusions. Their paper provides an analysis into

the intersection handling problem, and that the solutions learned by the network point out several shortcomings of the current rule-based methods they have given in the paper. They have also discussed the failures of the current deep reinforcement learning system they have used.

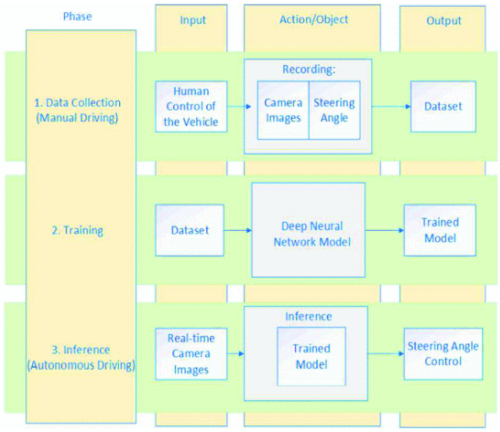
Ana I, Antonio and the other authors in [6] have worked on event based vision and deep learning for autonomous cars. Their approach is based on a deep neural network approach that unlocks the potential of event cameras and they have tested this on different motion estimation tasks, one of the tasks is to predict the vehicle's steering angle. They have used the sensor-algorithm combination. Their evaluation is done on a publicly available large scale event camera dataset of approx. 1000km. In their paper they have presented qualitative and quantitative analysis as to why event cameras allow robust steering predictions even in cases where traditional cameras fail like different illuminating conditions and during fast motions. They have also demonstrated the advantages of leveraging transfer learning from traditional to event-based vision and show that their approach outperforms state-of-the-art algorithms based on standard cameras.

3. BLOCK DIAGRAM

Our proposed model has three stages. Data collection, model training and model testing (Autonomous Driving). Data collection is done by using the Unity car simulator: Udacity. There are two modes in the car simulation software: Training mode and Autonomous Mode.



In the training mode, the car driver drives the vehicle along the path specified and the data set is acquired by capturing images from the moving car through out the drive. There are 3 mounted cameras on the car, specifically left mounted, right mounted and front mounted. The acquired data set will be used for preparing the neural network which will learn how to drive the vehicle independently with out human intervention. The images from the three separate cameras in the virtual world that are connected to the vehicle on the front hood, left mirror, and right mirror are collected continually while we record in the training mode, coupled with the steering angle for the different curves on the road. This strategy is otherwise called behavior cloning. 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.

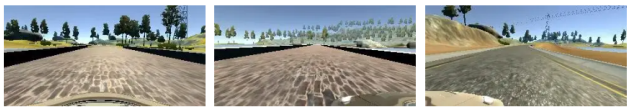


4. IMPLEMENTATION

Our goal is to use manually collected image data to teach the car to steer left and right based on conditions around. The ability of the model to drive the car is learned from cloning the behavior of a human driver.

4.1. IMAGE AUGMENTATION

Training data is obtained from examples of a human driving in the simulator, then fed into a deep learning network which learns the response (steering angle) for every encountered frame in the simulation. The car has 3 cameras on board - left, right and center camera. Cameras snapshot images of the road. We will use these images to train our neural network. Our model is trained to predict an appropriate steering angle for every frame while driving.



center | left | right cameras on track 1

center | left | right cameras on track 2

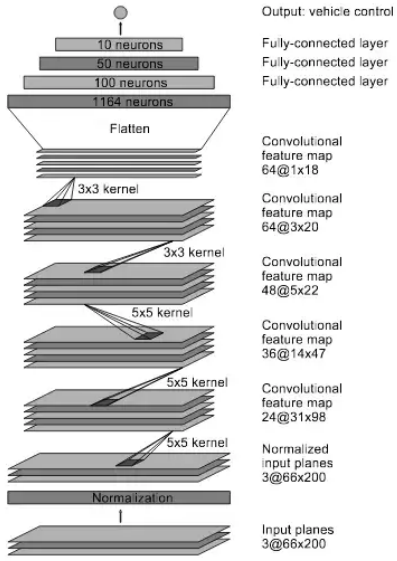
Data processing is done to allow our model to be able to easily work with raw data for training. The data processing is built into a generator (keras fit_generator) to allow for real-time processing of the data. While using the left and right images, we add and subtract 0.25 to the steering angles respectively to make up for the camera offsets. Since the original image size is 160 x 320 pixels, we randomly translate the image to the left or right and compensate for the translation in the steering angles with 0.008 per pixel of translation. We then crop a region of interest of 120x220 pixel from the image. In order to balance left and right images, we randomly flip images and change signs on the steering angles. The following figure shows the view from the left, right and center cameras after being jittered, cropped, and angles corrected. The right camera view has been flipped so it looks like a left camera image.

Different brightness instances are simulated by converting the image to HSV channel and randomly scaling the V channel.

The input image size is 160x320 and the image is vertically cropped at the top by removing half of the height which gives us an image of 80x320 resolution. The cropped image is normalized to make sure that the mean of the pixel distribution is zero. The cropped image is resized to 66x200 using tensorflow image.resize.

4.2. NVIDIA MODEL

We have used the Nvidia Convolution Neural Network model for training. This network has a total of 9 layers with 3 fully connected layers, 5 convolution layers and a normalization layer.



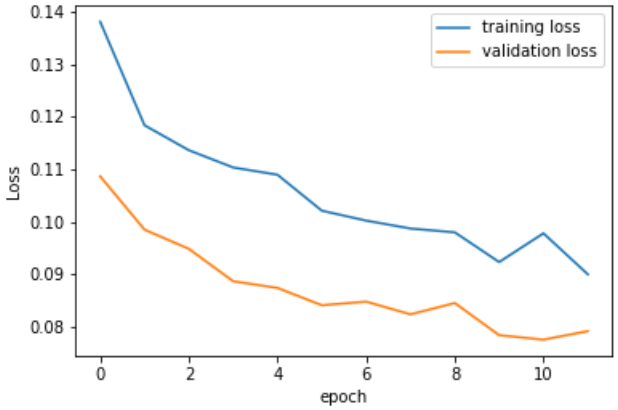
According to [2], Image normalization is done in the first layer & this enables normalization also to be accelerated via GPU processing. Convolution is done in the first three layers with 2x2 strides and a 5x5 kernel, non strided convolution is done with 3x3 kernel size in the last two convolution layers. The convolution layers are followed by three fully connected layers which output the steering angle.

Overfitting is reduced by using aggressive dropout (0.2) on all the layers and L2 regularization (0.001) on the first layer. Because of this the model could generalize to the second track. An Adam optimizer is used for stochastic gradient descent or optimization. The Adam optimizer requires no tuning as the learning rate is good. Early stop mechanisms are used during training to choose the best training model by monitoring the validation loss and stopping the training if the loss does not reduce in three consecutive epochs.

The first convolutional layer is with filter depth as 24 and filter size as (5,5) with (2,2) stride followed by ELU activation function. Moving on to the second convolutional layer with filter depth as 36 and filter size as (5,5) with (2,2) stride followed by ELU activation function. The third convolutional layer with filter depth as 48 and filter sizes as (5,5) with (2,2) stride followed by ELU activation function. Two convolutional layers are defined with filter depth as 64 and filter size as (3,3) and (1,1) stride followed by ELU activation function. Next step is to flatten the output from 2D to side by side. Here the first fully connected layer with 100 outputs is applied. Dropout with Dropout rate as 0.25 to combat overfitting are introduced here. Next, a second fully connected layer with 50 outputs are introduced. Then comes a third connected layer with 10 outputs. And finally, the layer with one output.

5. RESULTS & EVALUATION

We were able to successfully test the models performance on unknown tracks. Our model is trained on the data set acquired from the car simulator. The training and validation loss for the Nvidia model during model training phase was 0.098 and 0.0716 respectively.



With less training data, we observed that our model is able to perform better on 1st unseen track(compared to 2nd unseen track) which was quite similar to our training track.

To overcome this flaw we generated more training images using data augmentation technique and by driving the car in reverse direction using car simulator. Because of this we are able to add more variations in the training data set which is useful for behavioral cloning models. To overcome the overfitting problem we have considered implementing the drop out technique. Randomly selected neurons are ignored during the training phase of the model.

```
<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
```

In our training data set (11435), 35% of our images have 0 steering angle which was creating a biased behavior while training the model. To reduce this effect we removed around 20% of our total images (with 0° steering angle).

6. FUTURE WORK

In this project, only camera images from which steering angles were determined for the car to be driven on various tracks were used to train the model. If a real prototype model is to be created, then it will have more constraints and more features can be added. An example can be the detection of traffic lights. We can train the vehicle to stop at traffic lights and move accordingly. This can be achieved using CNN models for image recognition. Another example is lane end detection. The vehicle can make a U-turn and switch lanes by using sensors and estimating distance to detect objects, such as cars, the car can be stopped or have its speed reduced. VANETs is a technology based on IoV. Its primary role is to ensure uninterrupted access to resources such as the Internet for connectivity [7]. The rapid advancement of the Internet has increased the number of users, and VANETs can be used to satisfy their needs by accessing resources and staying connected to the Internet while on the move. It also had potential uses in other fields such as health and safety, intelligent transportation systems, and military systems, to name a few. Using VANET, we can cluster vehicles according to routing, mobility, and behaviors on the road, allowing us to keep an adaptive approach in monitoring traffic and pollution density. This information will also help the AVs make adaptive decisions for route selection. The advantages of using VANET for tracking, navigating, routing, and communication will become more apparent once vehicle-to-vehicle communication becomes approved by governments and all cars on the road adapt to this technology [8].

7. REFERENCES

[1] Singh, S. (2015). *Critical reasons for crashes investigated in the national motor vehicle crash causation survey* (No. DOT HS 812 115).

[2] 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*..

[3] 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

[4] 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.

[5] 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.

[6] 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).

[7] 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.

[8] Lee, M., & Atkison, T. (2021). Vanet applications: Past, present, and future. *Vehicular Communications*, *28*, 100310.