# Behavioral Cloning 2.0: Building upon Existing Work

Swapnil Ghosh
*School of Data and Sciences*
*Brac University*
Dhaka, Bangladesh
swapnil.ghosh@g.bracu.ac.bd

***Abstract*—Cloning someone's behavior involves physically making an exact copy of that person's actions. The goal is to use training data from the driver to teach a Convolution Neural Network (CNN) to drive in a manner that is analogous to that of the driver. NVIDIA published a study in which they trained CNN to transfer raw pixels from a single front-facing camera straight to steering instructions. This was accomplished with just one camera. Surprisingly, the findings were quite impressive, as the automobile was able to learn how to drive in traffic on local roads with or without lane markings, as well as on highways, with a minimal quantity of training data. Udacity will give us with a simulator, and we will utilize it. The front of the simulation vehicle is fitted with three cameras, one of which records video while the other two record the angle of the steering wheel in relation to the central camera. We will train the model in the same way as described in the paper, but with some modifications in order to make the model better.**

***Index Terms*—Convolution Neural Network (CNN), NVIDIA, Udacity**

## I. Introduction

An end-to-end learning system for self-driving cars was detailed in a recent research [1], in which a convolutional neural network (CNN) was trained to output steering angles given input pictures of the road ahead. This network is currently referred to as PilotNet. Images captured by a front-facing camera installed in a vehicle used for data collection were combined with the time-synchronized steering angle collected from a human driver to create the training data. The idea behind PilotNet was to do away with the need of manually coding rules and instead develop a system that can pick up new information simply by watching its surroundings. The first findings were positive; nevertheless, a significant amount of development work need to be done before such a system can drive itself without the assistance of a person. The simulator that Udacity gives us allows us to test and execute the model more effectively. I have made some adjustments to the preexisting PilotNet model in order to achieve some progress in the evaluation process. Here's the PilotNet 1 model described in the paper [1]

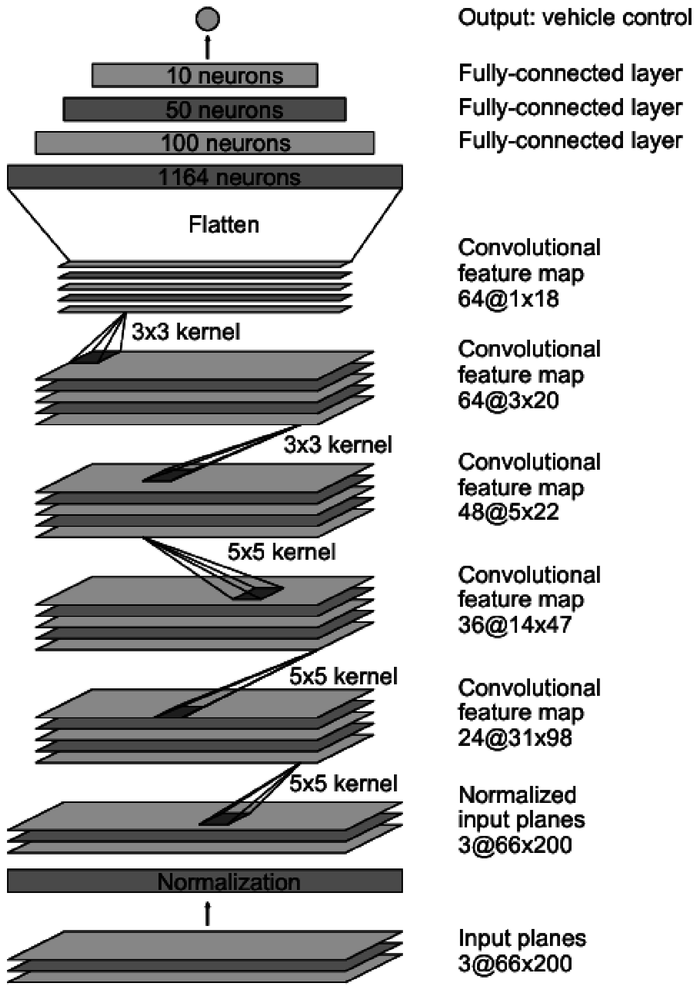


Fig. 1. CNN architecture. The network has about 27 million connections and 250 thousand parameters.

## II. Related Works

The 2016 article "End to End Learning for Self-Driving Cars" by Mariusz Bojarski et al. [1] introduces a novel method for training a convolutional neural network (CNN) to translate raw pixels from a single front-facing camera into driving commands without using human-designed features or predetermined rules. CNNs learn to drive on local roads with or without lane markings, freeways, and parking lots using an end-to-end method. Research suggests the model can accomplish this with minimal human driver data. The authors want to avoid "if, then, else" rules based on lane markers, guard rails, and other vehicles. To train the system, the authors directly link raw visual data with driving directions. Torch 7 and

NVIDIA DevBox trained their model. NVIDIA DRIVETM PX routed. 30 FPS. Scientists use mean squared error to teach the network to steer like humans. Nine layers—a normalization layer, five convolutional layers, and three fully connected layers—make up the network. Fully connected layers guide, while convolutional layers gather input image information. The authors provided the network input image YUV planes. The first three convolutional layers are strided, but the latter two are not. The article's vehicle control instructions are photo-based. Their end-to-end technique worked effectively in experiments. The machine learnt to drive in low-light, hazy, uneven, lane-marked conditions. Their method is scalable for large-scale application since it requires less human-sourced training data. ”End to End Learning for Self-Driving Cars” trains a CNN to turn raw pixels from a single front-facing camera into driving instructions. Method works. The paper provides a way for machines to drive without rules or human-designed features. The author's experiments reveal that their large-scale deployment approach is practicable and scalable, offering a potential path for self-driving car research.

”Playing Minecraft using Behavioural Cloning” [2] The MineRL 2019 competition required sample-efficient agents to play Minecraft using a dataset of real-world gaming and a fixed number of steps. Predicting human behavior through behavioural cloning solved this challenge. The authors observed that a simple algorithm might perform differently depending on when training ends. This article describes the authors' BC-based MineRL 2019 proposal and associated issues. The agent's performance variation during training, the effect of uniformly sampling the training dataset, the use of data augmentation to improve performance, and the agent's potential bias toward actions overrepresented in the dataset. The work highlights the unreliability of behavioural cloning and the necessity to report results variants, like RL research. Agents predict behavior via deep neural networks. Residual networks and direct features process image observations. Alternatives are sampled from each action's softmax activation probability. ReLU activation follows FixUp layer initialization. The model does not use LSTM or earlier frames. Since direct qualities affect action likelihood, inventory counts may reveal activities. Memory isn't needed to finish the challenge.

In ”Model-based Behavioral Cloning with Future Image Similarity Learning,” the authors Alan Wu et al. [3] provide a framework for visual imitation learning that allows robot action rules to be learned simply from expert examples without the need of actual robot trials. In a real-world setting, robot exploration and on-policy experimentation might often be costly or risky. This issue is solved using a novel method that enables generalised action cloning using future picture similarity and learns a future scene prediction model merely from a library of expert trajectories made up of unlabeled sample movies and actions. The robot gains the ability to foresee outcomes of actions visually. To choose the appropriate course of action, it compares the anticipated future picture to an expert image. The authors describe how we use future images for robot learning using a stochastic action-conditioned convolutional autoencoder. They tested models against several baseline techniques utilising a ground mobility robot in both simulated and actual situations, with and without obstacles.The model learns to produce a prior, zt, which changes depending on the input sequence, in addition to learning to input the current picture and action. The representation from before the future picture prediction is further concatenated with this. The prior is used to provide crisper visuals and improved modelling in unpredictable contexts. imagined future scenes doing various activities in simulations and real-world labs. Deterministic model with linear and convolutional state representation, top two rows of each environment. Stochastic model with linear and convolutional state representation, respectively, can be seen in the bottom two rows. Each row's central picture represents the current image, with neighbouring images to the left and right rotating by -5° and +5°, respectively. It can provide plausible pictures to train a critic V hat, which aids in choosing the best course of action, using the stochastic future image predictor. In the actual world and in simulated scenarios, the authors tested their critic model and future image prediction model.

By leveraging human demos to train a deep network, Kanervisto, Pussinen, and Hautamaki's paper [4] explores the viability of end-to-end behavioral cloning for video game playing. Twelve games are included in the study, including six modern games that were launched after 2010. The research demonstrates that while the agents cannot perform as well as humans, they can grasp basic dynamics and rules. The influence of human reaction delays on data quality is also highlighted by the authors, along with the significance of training data quantity and quality. Modern video games are said to use ViControl, a multi-platform application for recording and playing games, as a kind of behavioral cloning. The article emphasizes that game-specific engineering is still required for successful results despite behavioral cloning's benefits, such as not needing complex game adaptations and sparse training sessions. The neural network model is constructed using a convolutional neural network with three convolutional layers, one fully connected layer, and ReLU activations. It was trained with the Adam optimizer using a learning rate of 0.001 and an L2-normalization weight of 10-5 until the training loss did not decrease. The network's probability estimate is used to sample the final actions during evaluation, outperforming deterministically selecting the path of action with the highest likelihood. Overall, the article offers insightful information on the possibilities and difficulties of end-to-end behavioral cloning for playing video games.

This research ”Behavioral Cloning from Observation” [5] suggests a two-phase autonomous imitation learning method dubbed behavioural cloning from observation (BCO). In order to build a model, the agent must first gain experience in a self-supervised way. The model is then used to learn a certain task by watching an expert accomplish it without knowing the precise steps they took. The state-of-the-art generative adversarial imitation learning (GAIL) technique is compared to other imitation learning approaches, including the BCO approach, and it shows comparable task performance in a variety of simulation domains while displaying increased learning speed once expert trajectories are made available. The BCO approach seeks to increase performance in two key areas for autonomous agents: learning from observation alone and learning rapidly. The post-demonstration environment interaction in the modified version of the Behavioral Cloning Optimization (BCO) method described in the article enhances the learned model and imitation policy. After implementing the imitation policy in the environment for a brief period of time, the updated algorithm, known as BCO(alpha), changes the model and imitation policy using freshly observed state-action sequences. The number of contacts with the post-demonstration environment at each iteration is determined by the value of alpha, the user-specified parameter. It is possible to compute the total number of post-demonstration interactions needed by BCO(alpha) as a function of and the quantity of demos. The model may enhance the learnt imitation policy by appropriately assessing the demonstrator's actions when a nonzero alpha is used. By setting both alpha and T, the model-improvement iterations may be stopped early if the budget for post-demonstration interactions is fixed.

In the research ”Enhanced Behavioral Cloning Based self-driving Car Using Transfer Learning” [6] an unique end-to-end based VGG16 technique is provided, which is adjusted to forecast the steering angle based on the environmental limitations. This approach is intended to demonstrate the significance of transfer learning approach in the context of self-driving automobiles. The suggested method is then put up against NVIDIA and its pruned variants (which were reduced in parameter count by 22.2 percent and 33.85 percent, respectively, using a 1 x 1 filter). The training time is much shorter for the pruned architectures than for the baseline architecture since there are fewer parameters in them. Although just a portion of the network is taught for the pre-trained model, a considerable amount of computational time is saved without sacrificing performance when using the transfer learning technique. Transfer learning is a more effective approach to reduce training time when the tasks are comparable since the weights of the first few layers are identical and the latter layers include information that is relevant to the job. The VGG16 with transfer learning architecture surpassed other techniques with quicker convergence, according to experimental data.

The research ”Augmented Behavioral Cloning from Observation” [7] describes that in order to increase the efficiency of Imitation from Observation (IfO) in training an agent to imitate the behaviour of an expert, the study suggests a novel method termed Augmented Behavioral Cloning from Observations (ABCO). The suggested approach controls the data input into the inverse dynamics model, eliminating undesired local minima, by using attention models and a sampling mechanism. In order to increase sample efficiency and the calibre of the imitation policy model, the ABCO technique combines the inverse dynamics model with the policy model. The results demonstrate that, using either low-dimensional state spaces or raw photos as input, ABCO outperforms conventional behaviour cloning by using attention processes and a sampling method. The research demonstrates the superiority of the suggested methodology over state-of-the-art approaches using technical words like imitation learning, behavioural cloning, learning from demonstration, and deep learning.

”Robust Behavioral Cloning for Autonomous Vehicles using End-to-End Imitation Learning” by Tanmay Vilas Samak, Chinmay Vilas Samak, and Sivanathan Kandhasamy [8]proposes a pipeline for powerful behavioral cloning of a human driver. Data gathering, balancing, augmentation, preprocessing, and neural network training precede ego vehicle model deployment. The suggested approach provides longitudinal control signals based on the anticipated steering angle and other parameters. The research compares the recommended solution to the latest NVIDIA implementation and demonstrates its dependability. Trials show that the pipeline can clone three driving habits. The paper suggests creating explicit hardware or sim2real implementations, using a range of datasets, exploring various techniques to tackle generalization failure, and standardizing trials and assessment metrics for future research. The authors' lightweight, end-to-end behavioral cloning process is a major addition to autonomous driving. To increase model resilience, the research emphasizes choosing and balancing datasets, data augmentation, and preprocessing. The pipeline's connected control rules suggest it may outperform standard control approaches. The study proposes cloning human driving behavior in autonomous cars, and its suggestions for further research will progress the subject.

Felipe Codevilla et al.'s [9] ”Exploring the Limitations of Behavior Cloning for Autonomous Driving” discusses behavior cloning's limitations for autonomous driving. The study discusses behavior cloning's present limits and state-of-the-art. Behavior cloning is attractive for autonomous driving because it can learn complicated driving strategies from expert demonstrations without engineering. Distribution shift, covariate shift, and dataset bias restrict it. The authors use the CARLA simulator to test behavior clone models in different lighting and weather conditions, additional objects, and road layouts to highlight these limits. Data augmentation, domain adaptability, and diverse data sources are suggested to overcome behavior cloning's drawbacks. They quantify how well these methods handle behavior cloning's limitations. The article finds that behavior cloning is promise for autonomous driving but has limits that must be addressed for real-world implementation. The authors recommend creating more robust and dependable autonomous driving techniques that can manage real-world variable and complexity.

”Driver behavioral cloning using deep learning” [10] presents a deep learning-based strategy for educating self-driving cars. conduct cloning—learning a driver's conduct through demonstrations and applying it to new situations—is the writers' emphasis. The authors describe their model's architecture, which is built on a deep convolutional neural network (CNN) trained to anticipate human driver behavior from visual input. They then cover behavior copying data collection and processing problems such driver behavior variability and data imbalances. The authors suggest mimicking lighting, weather, and other environmental elements to enhance training data to solve these issues. They also detail balancing data to prevent the model from overfitting to any driving style or environment. The authors test their approach using real-world driving video and compare it to baseline techniques. Their strategy exceeds baseline approaches in accuracy and generalization to new contexts. The research introduces a unique deep learning technique to behavior cloning and discusses data collection and processing problems. The suggested strategy shows promise for self-driving car development.

The study ”Behavior Cloning for Autonomous Driving using Convolutional Neural Networks” by Wael Farag and Zakaria Saleh [11] offers a CNN-based safe steering controller named ”BCNet” for autonomous driving. A front-facing camera and steering orders from an experienced driver operating in traffic and urban roads provide training data. The obtained data trains the suggested CNN for behavioral cloning. After numerous experiments, the BCNet features a deep seventeen-layer design. Adam's optimization technique trains BCNet. The study describes the creation, training, and image processing pipeline. After numerous simulations, the suggested technique clones the driving behavior in the training data set. CNN-based lane and road following may be learned without human dissection into road or lane marker recognition, semantic abstraction, route planning, and control. One or two tracks of training data are enough to teach the automobile to drive safely on several tracks. Steering alone trains the CNN to recognize road elements. The authors examine the approach's drawbacks and suggest improvements for future study. This application requires an extensive training data pre-processing pipeline. This application requires data quality above quantity, as the authors underline. This approach advances completely autonomous autos. The article makes a strong argument for CNN behavior cloning in autonomous driving.

The research ”Data Augmented Deep Behavioral Cloning for Urban Traffic Control Operations Under a Parallel Learning Framework” [12] presents a data augmented deep behavioral cloning (DADBC) approach to mimic traffic engineers' time-consuming and expert problem-solving abilities in traffic signal control (TSC) operations. The parallel learning (PL) framework uses machine learning to solve complicated system decision-making challenges. DADBC employs a generative adversarial network (GAN) and the deep behavioral cloning (DBC) model to learn traffic engineers' control techniques. The authors stress the need for a software-aided tool that can learn from human specialists to speed up traffic control and management in vast metropolitan traffic networks. They recommend a data-driven approach over analytical modeling to reflect human knowledge's stochastic and uncertainty features for traffic management operations. The study briefly explains the generative adversarial model and deep behavioral cloning approach and summarizes TSC system research. Using actual manipulation data from Hangzhou, China, the authors demonstrate that the suggested technique can emulate complicated human behaviors in intervening traffic signal control operations to increase urban traffic efficiency. DADBC allows the control system to deliver timely and high-performance control methods. The report finishes with further research. The research proposes a unique method to mimic traffic engineers' problem-solving abilities in TSC operations and improve traffic control systems in big metropolitan traffic networks. The paper's data-driven methodology and machine learning in a parallel learning framework are important. The validation findings imply that the suggested solution might overcome TSC operating issues.

”Behavioral Cloning for Lateral Motion Control of Autonomous Vehicles using Deep Learning” by Sharma et al. [13] covers the issues of constructing an end-to-end model for autonomous driving with an emphasis on autonomous lateral control. The authors emphasize autonomous lateral control in self-driving automobiles and the limits of standard image processing. A convolutional neural network outputs steering angles from road photos in their end-to-end learning technique. Path planning and following algorithms are unnecessary since the model is taught to match human driving behavior. The authors train a deep neural network on 10 hours of driving data from two tracks using the TORCS simulation environment to prove their method works. With four hours of data from one track, the end-to-end model maintained lanes and completed laps on multiple courses. End-to-end learning and behavioral cloning may drive independently in new and unforeseen settings. The model guided the car successfully 89.02 percent of the time on single-lane unfamiliar tracks and 96.62 percent on multilane tracks. The work analyzes data collecting and training systems and installs all components needed to train an end-to-end model, adding to the literature. The writers explore training issues and remedies. The article solely covered lateral control and did not address longitudinal control. They advise training the network using vehicle speed and steering angles for autonomous longitudinal control. The article sheds light on deep learning and neural network training software for autonomous driving. The study shows that end-to-end learning and behavioral cloning can drive autonomously in new and unknown scenarios, and its detailed analysis of data acquisition and training systems is useful for autonomous driving researchers and engineers.

Nelson Fernandez Pinto and Thomas Gilles' [14] ”Enhanced Behavioral Cloning with Environmental Losses for Self-Driving Vehicles” covers safe path planning. While trained route planners may mimic human driving behavior and give quick inference, they cannot handle complicated driving circumstances by simply imitating expert observations. Predictions beyond drivable zones might also be harmful. Social loss and Road loss loss functions describe dangerous social interactions in route planning. These losses repel non-drivable regions. Backpropagation reduces training cost for predictions around these locations. This approach adds environment input to supervised learning. On a large-scale urban driving dataset, the authors showed that the agent learns to drive like humans while improving safety measures. The study emphasizes safe route planning in self-driving cars, especially in complex and dynamic metropolitan contexts. The authors highlight that autonomous automobile technology faces substantial obstacles, and self-driving perception and planning systems must be reliable. In multi-agent urban situations, model-based planners are generally suboptimal. Due of their short inference time and ability to imitate nuanced human driving behavior, learned planners—particularly behavioral cloning—have garnered attention. The research finds that their suggested behavioral cloning strategy reduces collisions and overlaps with non-drivable zones. Inference has persistent impacts without risky driving teaching instances. This article contributes to safe route planning in self-driving cars and calls for further investigation.

”A Deep Network System for Simulated Autonomous Driving Using Behavioral Cloning” by Patachi, Leon, and Logofătu [15] describes a technique for training a convolutional neural network (CNN) to learn vehicle behavior using simulator data. The authors show that the CNN can learn to drive on a defined strip road with little driving experience through suitable driving simulations in a game. The network dynamically learns how to represent processing stages including road feature identification, speed, and track location. The study explores testing driving assistance functions like auto emergency braking and cruise control using virtual simulation platforms. With the rush to deploy completely driverless automobiles, these platforms are increasing appeal, the authors say. They explore simulators like CarMaker, Udacity-self driving car, and CARLA that can simulate automobiles, traffic signs, pedestrians, sensors, and weather. The writers also discuss behavioral cloning, which transfers human talents to a computer program. They explain how a learning system may utilize a log of human subject task records to imitate the desired behavior. This method may be used to build complex automated control systems that traditional control theory cannot handle. Data collection and processing, convolutional neural network design, and activation functions are summarized in the paper's conclusion. The exponential linear unit (ELU) activation function improves learning. They show experimental data and discuss autonomous driving research.

## III. Research Methodology

### A. Data Collection:

The simulator features two lanes, the first of which is rather straightforward with fewer, smaller bends, and the other of which is challenging with many, winding turns and steep slopes. Training data from both tracks will be used. We'll maintain lane center while driving in both lanes. Each of us will do 3 laps in the car. We'll drive one lap in each of the lanes while attempting to drift to the outside and keep the lane in the middle. This will provide us with training data for our model adjustments.

### B. Data Augmentation:

Randomly flip several photos and set the steering angle to -steering_angle. Randomly move the graphics horizontally and vertically and modify the steering angle. Road shadows include trees, posts, etc. We'll shadow training photos. Randomly brighten photos. Some training picture 2 outputs after augmentations are below.

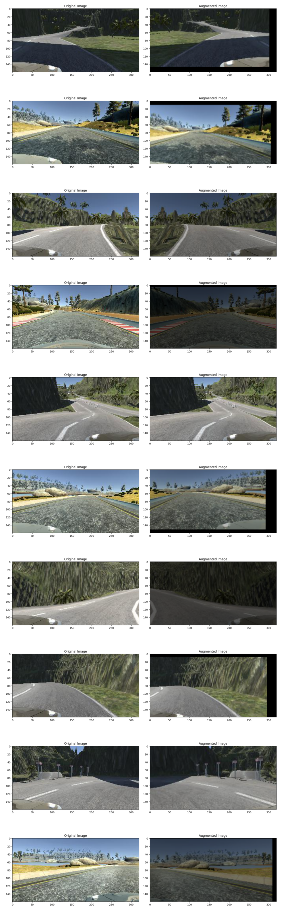


Fig. 2. Original Image vs Augmented Images

### C. Pre-Processing:

The training pictures are of size 160*320*3, whereas the input images are 66*200*3. Additionally, we must transform input photos from RGB to YUV color format. Additionally, the presence of mountains or a vehicle bonnet at the bottom of the picture is not necessary for training purposes. So, from the input photos, we will crop the top 40 and bottom 20 pixel rows. As part of the pre-processing, we will also scale the cropped picture to 66*200*3 and change the color space to YUV.

### D. Model Architecture:

The NVIDIA paper [1] on mapping raw images to driving signals inspired my Neural Network Architecture. 9 layers—a normalization layer, 5 convolution layers, and 3 fully connected layers—make up the network. Normalization layer normalizes image. Normalization in the network may be changed with the network design and expedited with GPU processing. 5 convolution layers extract picture features. The first three convolutions are strided with 2x2 strides and a 5x5

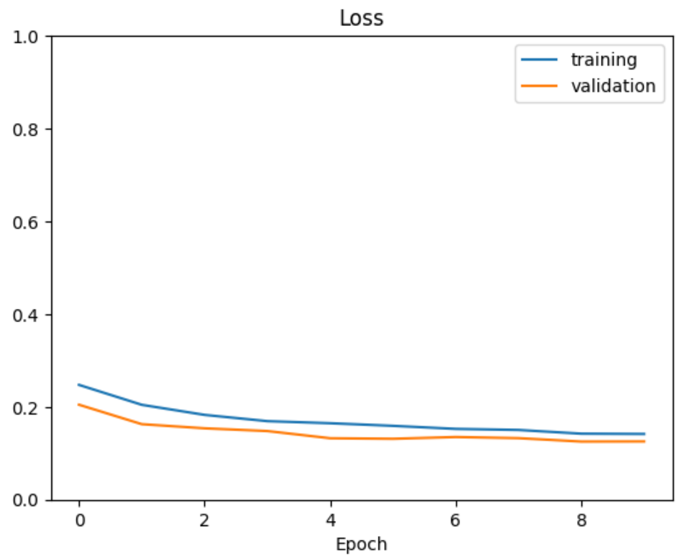
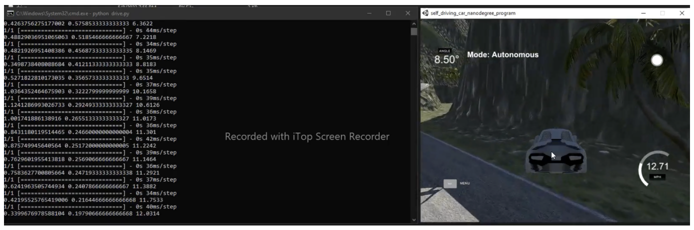
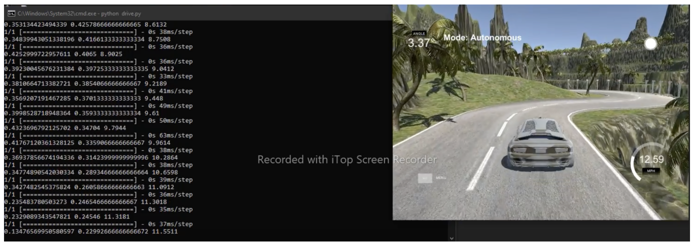


Fig. 3. Training result


Recorded with iTop Screen Recorder
self_driving_car_nanodegree_program
ANGLE
8.50°
Mode: Autonomous
12.71

```
C:\Windows\System32\cmd.exe - python  drive.py
0.4263756275177002 0.5758533333333333 6.3622
1/1 [==============================] - 0s 44ms/step
0.48829036951065063 0.5185466666666667 7.2218
1/1 [==============================] - 0s 34ms/step
0.4821926951408386 0.4568733333333335 8.1469
1/1 [==============================] - 0s 35ms/step
0.3498738408088684 0.4121133333333333 8.8183
1/1 [==============================] - 0s 35ms/step
0.5271822810173035 0.3565733333333333 9.6514
1/1 [==============================] - 0s 37ms/step
1.0364352464675903 0.3222799999999999 10.1658
1/1 [==============================] - 0s 39ms/step
1.1241286993026733 0.2924933333333327 10.6126
1/1 [==============================] - 0s 36ms/step
1.001741886138916 0.2655133333333327 11.0173
1/1 [==============================] - 0s 36ms/step
0.8431180119514465 0.2466000000000004 11.301
1/1 [==============================] - 0s 42ms/step
0.875749945640564 0.2517200000000005 11.2242
1/1 [==============================] - 0s 39ms/step
0.7629601955413818 0.2569066666666667 11.1464
1/1 [==============================] - 0s 36ms/step
0.7583627700805664 0.2471933333333338 11.2921
1/1 [==============================] - 0s 37ms/step
0.6241963505744934 0.2407866666666667 11.3882
1/1 [==============================] - 0s 34ms/step
0.42195525765419006 0.2164466666666668 11.7533
1/1 [==============================] - 0s 40ms/step
0.3399676978588104 0.1979066666666668 12.0314
```

Fig. 4. Simulation run on the existing model and dataset: Crashed Car



```
C:\Windows\System32\cmd.exe - python  drive.py
0.353134423494339 0.4257866666666665 8.6132
1/1 [==============================] - 0s 38ms/step
0.34839943051338196 0.4166133333333334 8.7508
1/1 [==============================] - 0s 36ms/step
0.4252999722957611 0.4065 8.9025
1/1 [==============================] - 0s 36ms/step
0.39230045676231384 0.3972533333333335 9.0412
1/1 [==============================] - 0s 33ms/step
0.3810664713382721 0.3854066666666667 9.2189
1/1 [==============================] - 0s 41ms/step
0.3569207191467285 0.3701333333333333 9.448
1/1 [==============================] - 0s 49ms/step
0.3998528718948364 0.3593333333333334 9.61
1/1 [==============================] - 0s 50ms/step
0.4323696792125702 0.34704 9.7944
1/1 [==============================] - 0s 63ms/step
0.41767120361328125 0.3359066666666667 9.9614
1/1 [==============================] - 0s 38ms/step
0.36937856674194336 0.3142399999999996 10.2864
1/1 [==============================] - 0s 38ms/step
0.34774890542030334 0.2893466666666664 10.6598
1/1 [==============================] - 0s 39ms/step
0.3427482545375824 0.2605866666666663 11.0912
1/1 [==============================] - 0s 36ms/step
0.235483780503273 0.2465466666666667 11.3018
1/1 [==============================] - 0s 35ms/step
0.2329089343547821 0.24546 11.3181
1/1 [==============================] - 0s 37ms/step
0.13476569950580597 0.22992666666666672 11.5511
```

Fig. 5. Simulation run on the tweaked model and new created dataset: Car not Crashed

kernel. Non-strided 3x3 kernel convolutions provide the final two convolution layers. Three full-connected layers follow the convolution layers to provide the output steering value. Keras and Tensorflow power the model. Adam Optimizer is used to optimize the learning. This optimizer automatically adjusts learning rate during training.

### *E. Tweaking the Model:*

Initially the model was under-fitting and the car showed a bias towards straight driving, failing to turn at curves. By adding recovery laps, more data samples using data augmentation, a drive on Track2, the bias towards straight drive was eliminated. The vehicle was making erroneous turns or straying into the track's borders. I noticed that the training losses were far lower than the validation losses. I determined that over-fitting was the cause of the issue. By including dropout layers after all layers with a drop probability of 0.5, this was reduced to a minimum. I attempted a model with "relu" activation instead of "elu." Dropout after Flatten layer regularized. "Relu" non-linearity after dense layer. Model training took 15 epochs. Here is the result 3.

## IV. Results

I utilized the already existing model, and I used the already existing dataset, and I obtained a result 4 where we can see that Track2 has a vehicle that crashes at the beginning, and it is unable to recover. This takes place on every run. On Track1, though, the vehicle does not have any issues. After making a few adjustments to the previously developed model, I trained it using a dataset that I had constructed myself. After putting the model through its paces, I was able to get a result 5 showing that the automobile performs well on both Track 1 and Track 2.

## V. Future Work

Although the model has made tremendous strides in its capacity to detect and react to various driving conditions, it is still not entirely capable of distinguishing between people and other road users or animals. This is mainly because situations involving live things are not often included in the datasets used to train these vehicles. However, the model's capacity to distinguish between people and other animals on the road would significantly advance the field. If the model is able to recognize and react to live things on the road, the safety of bikers, pedestrians, and cars might all be significantly increased.

## VI. Conclusion

This Udacity project is very difficult for those who are just starting out in the field of machine learning. A deep learning model that can drive independently on a simulated track was built by utilizing data taken from human behavior. This data was used to train the model. The model was able to learn by replicating human behavior, and it was able to generalize its response to a new test track. The model consists of five layers of convolution and three additional layers that are completely coupled. The model now simply controls the angle at which the steering wheel is turned, but it may be expanded to include control the throttle and the brake.

## Acknowledgment

Our faculty, Meem Arafat Manab sir and Joyanta J. Monda sir, deserve my sincere gratitude. My research's direction and rigor and quality were greatly influenced by their advice, knowledge, and support.

## References

[1] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," 2016.

[2] A. Kanervisto, J. Karttunen, and V. Hautamäki, "Playing minecraft with behavioural cloning," 2020.

[3] A. Wu, A. Piergiovanni, and M. S. Ryoo, "Model-based behavioral cloning with future image similarity learning," 2019.

[4] A. Kanervisto, J. Pussinen, and V. Hautamäki, "Benchmarking end-to-end behavioural cloning on video games," 2020.

[5] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," 2018.

[6] U. Sumanth, N. S. Punn, S. K. Sonbhadra, and S. Agarwal, "Enhanced behavioral cloning-based self-driving car using transfer learning," in *Data Management, Analytics and Innovation: Proceedings of ICDMAI 2021, Volume 2*. Springer, 2021, pp. 185–198.

[7] J. Monteiro, N. Gavenski, R. Granada, F. Meneguzzi, and R. Barros, "Augmented behavioral cloning from observation," 2020.

[8] T. V. Samak, C. V. Samak, and S. Kandhasamy, "Robust behavioral cloning for autonomous vehicles using end-to-end imitation learning," *SAE International Journal of Connected and Automated Vehicles*, vol. 4, no. 3, aug 2021. [Online]. Available: https://doi.org/10.4271%2F12-04-03-0023

[9] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," 2019.

[10] J. Kocić, N. Jovičić, and V. Drndarević, "Driver behavioral cloning using deep learning," in *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*, 2018, pp. 1–5.

[11] W. Farag and Z. Saleh, "Behavior cloning for autonomous driving using convolutional neural networks," in *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, 2018, pp. 1–7.

[12] X. Li, P. Ye, J. Jin, F. Zhu, and F.-Y. Wang, "Data augmented deep behavioral cloning for urban traffic control operations under a parallel learning framework," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 5128–5137, 2022.

[13] S. Sharma, G. Tewolde, and J. Kwon, "Behavioral cloning for lateral motion control of autonomous vehicles using deep learning," in *2018 IEEE International Conference on Electro/Information Technology (EIT)*, 2018, pp. 0228–0233.

[14] N. F. Pinto and T. Gilles, "Enhanced behavioral cloning with environmental losses for self-driving vehicles," 2022.

[15] A.-I. Patachi, F. Leon, and D. Logofătu, *A Deep Network System for Simulated Autonomous Driving Using Behavioral Cloning*, 05 2019, pp. 235–245.

## Appendix

Behavioral Cloning: Autonomous Car Simulator

How to Simulate:

Step_1:

Available Game Builds (Precompiled builds of the simulator) Term 1 Instructions: Download the zip file, extract it and run the executable file.

Version 2, 2/07/17

Linux Mac Windows

Version 1, 12/09/16

Linux Mac Windows 32 (Windows 64)

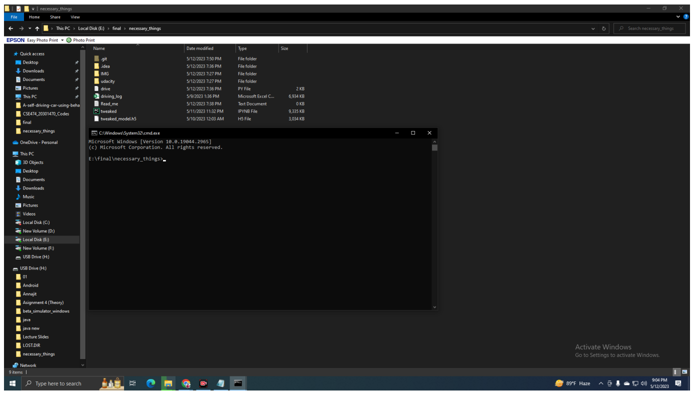
Step_2 : Download the files from the provided link and extract it in an empty folder. Link: Drive or github

Step_3: Open Command Prompt on the folder. 6

Step_4: Type: "udacity\Scripts\activate" and press enter. This will activate the virtual environment.7

Step_5: Then open the Udacity Simulator and run the simulator. When the simulator is open select a map and click the "Autonomous Driving" option.8

Step_6: Lastly, on the cmd type: "python drive.py" and press enter. You will see the server gets Connected and the car starts to drive autonomously.9



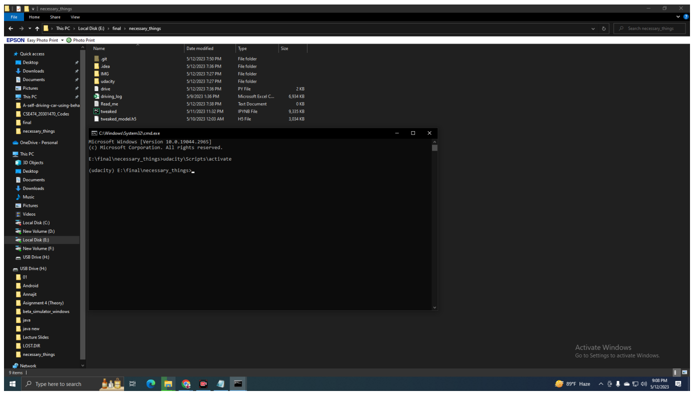
```
This PC > Local Disk (E:) > final > necessary_things

Name                 Date modified        Type                Size
.git                 5/12/2023 7:50 PM    File folder
.idea                5/12/2023 7:36 PM    File folder
IMG                  5/12/2023 7:27 PM    File folder
udacity              5/12/2023 7:27 PM    File folder
drive                5/12/2023 7:36 PM    PY File             2 KB
driving_log          5/9/2023 1:36 PM     Microsoft Excel C...  6,934 KB
Read_me              5/12/2023 7:38 PM    Text Document       0 KB
tweaked              5/11/2023 11:32 PM   IPYNB File          9,335 KB
tweaked_model.h5     5/10/2023 12:03 AM   H5 File             3,034 KB
```

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.2965]
(c) Microsoft Corporation. All rights reserved.

E:\final\necessary_things>
```

Fig. 6. Command Prompt



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.2965]
(c) Microsoft Corporation. All rights reserved.

E:\final\necessary_things>udacity\Scripts\activate

(udacity) E:\final\necessary_things>
```

Fig. 7. Virtual Environment

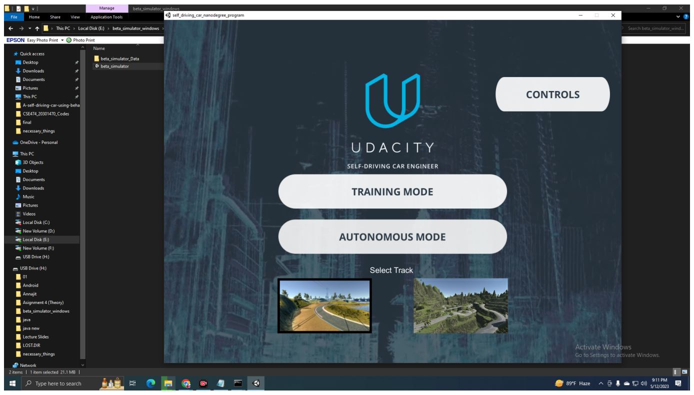
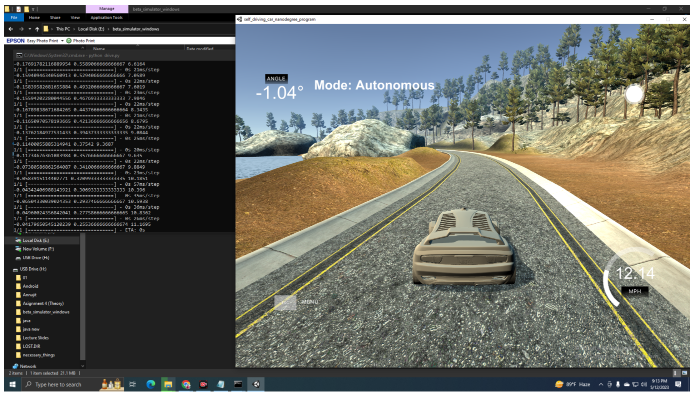


Fig. 8. Simulator


C:\Windows\System32\cmd.exe - python drive.py
self_driving_car_nanodegree_program
ANGLE
-1.04°
Mode: Autonomous
12.14
MPH

```
-0.17691782116889954 0.5589066666666667 6.6164
1/1 [==============================] - 0s 21ms/step
-0.15940946340560913 0.529406666666666 7.0589
1/1 [==============================] - 0s 22ms/step
-0.15839582681655884 0.4932066666666667 7.6019
1/1 [==============================] - 0s 23ms/step
-0.15594202280044556 0.4676933333333333 7.9846
1/1 [==============================] - 0s 22ms/step
-0.16789838671684265 0.44376666666666664 8.3435
1/1 [==============================] - 0s 21ms/step
-0.11650970578193665 0.42136666666666656 8.6795
1/1 [==============================] - 0s 22ms/step
-0.13762184977531433 0.39437333333333335 9.0844
1/1 [==============================] - 0s 25ms/step
-0.11400055885314941 0.37542 9.3687
1/1 [==============================] - 0s 20ms/step
-0.11734676361083984 0.3576666666666667 9.635
1/1 [==============================] - 0s 22ms/step
-0.07380586862564087 0.3410066666666667 9.8849
1/1 [==============================] - 0s 23ms/step
-0.0583915114402771 0.32099333333333335 10.1851
1/1 [==============================] - 0s 57ms/step
-0.04342406988143921 0.3069333333333333 10.396
1/1 [==============================] - 0s 35ms/step
-0.06504330039024353 0.2937466666666667 10.5938
1/1 [==============================] - 0s 36ms/step
-0.04960024356842041 0.27758666666666665 10.8362
1/1 [==============================] - 0s 26ms/step
-0.04179650545120239 0.25536666666666674 11.1695
1/1 [==============================] - ETA: 0s
```

Fig. 9.   Drive