CarSim: A Randomized Multi-Model Approach

Jun Hu
Cornell University
Ithaca, NY
jh2829@cornell.edu

Abstract

The integration of deep learning into autonomous driving systems represents a pivotal advancement in direct sensory-to-command mapping. Building upon the foundational work of LeCun et al. [1], this paper explores multiple RL and image-based approaches that significantly enhance the performance of autonomous driving agents in Udacity's Self-Driving Car Simulator and the gym highway-env. Our methodology focuses on advanced data augmentation and fine-tuning of learning algorithms to improve system adaptability and reliability across diverse driving conditions. Comprehensive testing in simulated testing environments validates our approach, demonstrating substantial improvements in the robustness and decision-making capabilities of driving agents. The enhanced model not only excels in standard scenarios but also shows superior fault tolerance in unpredictable conditions.

https://github.com/jjunhu/autonomous_driving_sim

1 Introduction

The domain of autonomous vehicles has witnessed transformative changes with the advent of deep learning technologies, particularly through the application of Convolutional Neural Networks (CNNs). The shift from traditional modular architectures—where perception, decision-making, and control are handled separately—to an integrated end-to-end learning approach has been revolutionary. Pioneering studies, such as those by LeCun et al. [1] and Bojarski et al. [2], have demonstrated the potential of CNNs to learn driving commands directly from raw pixel data. This research extends foundational models, focusing on enhancing the performance and reliability of autonomous agents in complex simulated environments like Udacity's Self-Driving Car Simulator and the gym highway env.

Figure 1: Udacity's Self-Driving Car Simulator

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

2 Background and Related Work

In this study, we employ a sophisticated suite of models and simulation environments aimed at advancing our understanding of autonomous driving behaviors under various conditions.

2.1 Behavior Cloning and CNN Network

Behavior Cloning (BC) is a form of supervised learning in autonomous driving where the goal is to mimic human driving behavior. The approach involves training a Convolutional Neural Network (CNN) to predict steering angles from direct sensor inputs, typically images captured from multiple camera angles. This method was first notably implemented in autonomous driving by Pomerleau [3] and has since been refined to enhance its performance and applicability. The neural network architecture often includes multiple convolutional layers that help in extracting features from the input images, followed by several fully-connected layers that predict the driving commands.

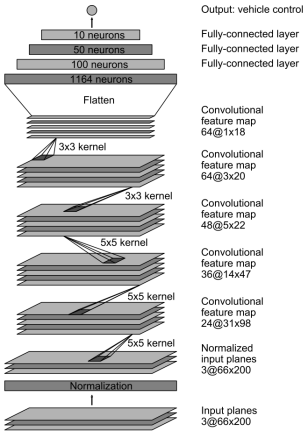

Figure 2: CNN Architecture Used for Behavior Cloning in Udacity Car Sim

Figure 2 illustrates a typical CNN used in these systems, where the input layer receives normalized images from vehicle-mounted cameras. The convolutional layers apply various filters to detect features such as lane markings and road edges. The fully-connected layers then interpret these features to produce a steering command. The training process involves adjusting the weights of the network through backpropagation to minimize the difference between the predicted and actual steering commands, as shown in Figure 3. Note we experimented with adding and deleting the dropout layers; although the dropout layers are shown here, we decided against it due to its poor performance impact. The operational mechanism of such a network is further depicted in Figure 4, where inputs from multiple cameras are processed to generate a comprehensive response that guides vehicle steering.

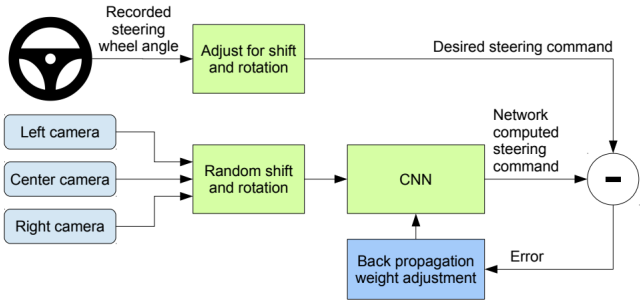

Figure 3: Training Process of CNN Used in Behavior Cloning

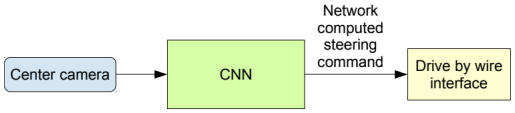

Figure 4: Operational Mechanism of CNN in Testing Environment

2.2 Udacity's Driving Simulator

Udacity's Self-Driving Car Simulator is a tool for training and evaluating autonomous driving algorithms. During training, the simulator captures images from three perspectives—left, center, and right cameras—mimicking the human visual field. These images serve as inputs for the CNN, which outputs vehicle control commands including steering angle, throttle, reverse, and speed. As detailed in the notebook, the simulator allows for comprehensive scenario testing, enabling researchers to refine algorithms under controlled conditions that simulate real-world driving complexities.

2.3 Highway-env and StableBaselines

The highway-env is a simulation environment designed for the development and validation of autonomous driving policies. This environment, particularly the 'highway-fast-v0' configuration, is employed to test various reinforcement learning algorithms. In this paper, we focus on three algorithms from the Stable Baselines library: Deep Q-Network (DQN) with Multi-Layer Perceptron (MLP), Proximal Policy Optimization (PPO), and DQN with CNN. The environment has been configured to accommodate various scenarios, including multiple agents, increased lane counts, and higher vehicle densities, to thoroughly evaluate the algorithms' performance under increased traffic complexities.

3 Problem

One of the principal challenges in the development of autonomous driving technologies is the inherent limitations posed by the availability and diversity of training data. Autonomous driving systems, particularly those developed through behavior cloning, rely heavily on large and varied datasets to learn and generalize driving behavior effectively. However, the complexity and cost associated with collecting extensive real-world driving data restrict the breadth and depth of training scenarios available, leading to significant constraints on system performance.

In the context of Udacity's car simulator, traditional behavior cloning algorithms exhibit suboptimal performance, especially evident after numerous training iterations. Despite advances in computational techniques and algorithm design, these systems struggle to achieve satisfactory reliability and adaptability when trained on limited datasets. This shortfall is primarily due to the algorithms' inability to extrapolate beyond their training conditions, rendering them less effective in unanticipated or novel driving environments.

This research seeks to address these challenges by exploring innovative approaches to augment training data and enhance algorithm robustness. By employing advanced data augmentation techniques and refining learning processes, we aim to significantly improve the performance of behavior cloning algorithms within Udacity's simulator framework. Our goal is not only to optimize these systems for better performance under standard conditions but also to elevate their capacity to handle diverse and unpredictable scenarios, thus bridging the gap between simulated training and real-world applicability.

This investigation is critical, as highlighted by Bojarski et al. [2] and Pan et al. [4], who have pointed out the limitations of existing datasets and learning strategies in training robust autonomous systems. Enhancing the ability of these systems to perform under varied conditions will be a significant step forward in the reliable deployment of autonomous vehicles.

4 Approach

4.1 Data Collection

One round of expert data collection is conducted for each track of the Udacity simulator. Each data point consists of center, left, right images, and steering angle, throttle, reverse, and speed. We collected overall 2095 data points.

4.2 Data Cleaning

In our dataset comprising 2095 data points, a significant imbalance was noted, with over 1300 data points featuring a steering angle of zero. This skew in the dataset could potentially bias the training process of our models, leading to overfitting and poor generalization on other steering angles.

To address this issue and enhance the robustness of our model, we implemented a data balancing technique. Specifically, we capped the maximum number of data points for any single steering angle at 400. This constraint was applied uniformly across all steering angles to ensure a more equitable distribution of data.

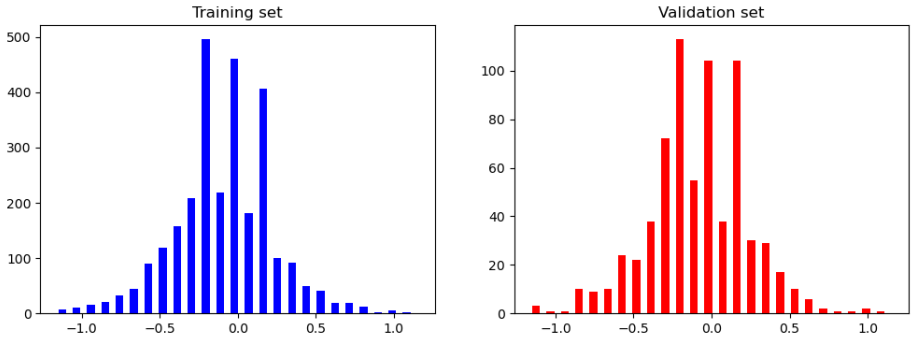

Figure 5: Train vs. Validation Data Distribution

4.3 Data Augmentation

Each data point originally contains three images; we restructured this by converting each image into a separate data point. This modification effectively triples the size of our dataset, providing more samples for model training and validation.

In terms of image preprocessing, our approach involves several steps aimed at optimizing the images for better model performance. Figure 5 shows an example. Here's a breakdown of the preprocessing steps applied to each image:

- **Cropping**: The image is cropped to remove irrelevant parts, specifically the top 60 to 135 pixels, focusing on the road and immediate surroundings which are more relevant for driving decisions.
- **Color Space Conversion**: We convert the image from RGB to YUV color space. This conversion is beneficial for driving models as it separates luminance from chrominance, which can be helpful in identifying textures and edges more effectively.
- **Gaussian Blurring**: To reduce high-frequency noise, a Gaussian blur is applied with a kernel size of (3, 3). This smoothing helps in reducing detail and noise, making some features more prominent for the driving model.
- **Resizing**: The image is resized to 200x66 pixels. This standardization ensures that the input size is consistent for the neural network, facilitating faster processing without sacrificing necessary details.
- **Normalization**: Finally, pixel values are normalized by dividing by 255. This step converts pixel values to a range of 0 to 1, enhancing the model's convergence speed during training by ensuring that input feature magnitudes are similar.

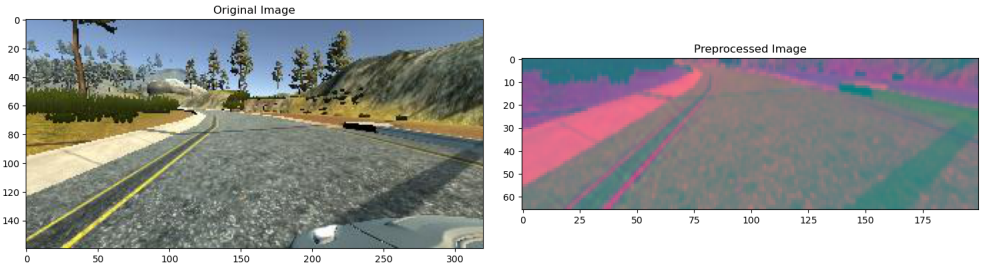

Figure 6: Image Proprocessing

4.4 Model Architecture

The primary methodology involves the utilization of behavior cloning within a controlled simulation environment. Refer to 2.1 for an in-depth look of the model. Training is done for 10 epochs, with a batch size of 100.

5 Results

5.1 Baseline Model

The training and validation loss over epochs are depicted in 8. Initially, the training loss demonstrates a sharp decrease, indicative of the model quickly adapting to the patterns within the training set. However, the validation loss plateaus early and remains approximately constant, suggesting an overfitting scenario where the model fails to generalize beyond the training data. This discrepancy between training and validation performance highlights the model's limitations in adapting to unseen scenarios.

To further understand the practical implications of these training results, we analyzed the model's behavior in simulated driving tests. Two tracks were used to evaluate the model's ability to handle real-time driving tasks, focusing on speed regulation and obstacle navigation. As shown in 7 and 20, the vehicle initially accelerates to the speed limit but demonstrates a significant variance in speed, particularly when approaching and maneuvering around obstacles. The model's response to obstacles is delayed, leading to sharp decelerations and, in some instances, complete stops, indicating a struggle to anticipate and smoothly navigate changes in the driving environment.

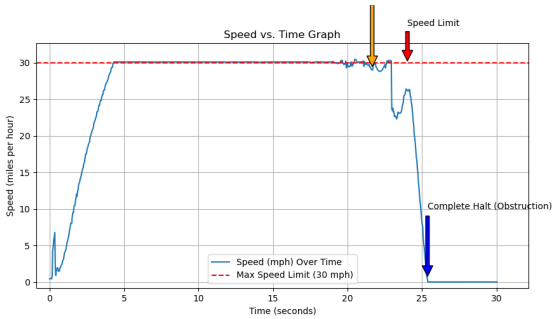

Figure 7: Performance on Track 1 without Data Augmentation

5.2 Improved Model

Recognizing the limitations of the baseline model trained without data augmentation, we introduced a series of randomization techniques aimed at enhancing the robustness and generalization ability of the behavior cloning model. Data augmentation serves as a pivotal strategy in our approach, artificially enlarging the training dataset and simulating a wider array of driving conditions that the model may not have encountered during the initial training phase.

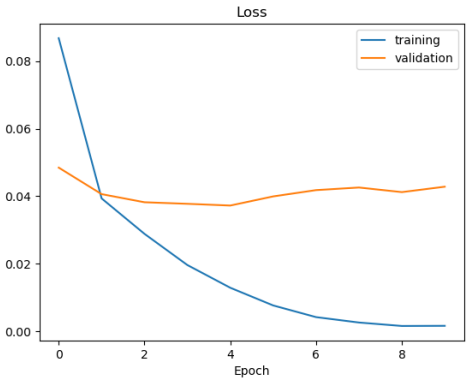

Figure 8: Epoch vs. Loss Training Curve for Baseline Model

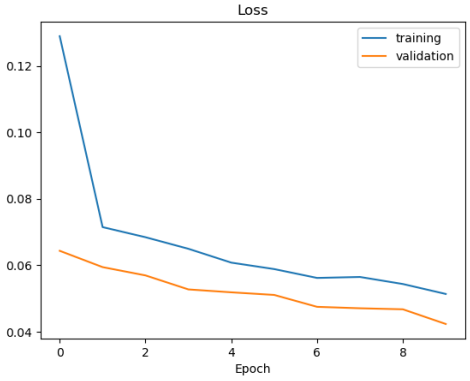

Figure 9: Epoch vs. Loss Training Curve with Data Augmentation

5.2.1 Augmentation Techniques

To infuse variability into the training process and thereby enhance model performance, we implemented the following data augmentation techniques, each designed to mimic real-world variability in driving scenarios:

- **Zoom**: To simulate the effect of varying distances from objects and road features, a random scaling between 1 and 1.3 times the original size was applied. This variability helps the model learn to adjust to objects at different distances. (16)
- **Pan**: Random translations in the x and y directions between -10% and 10% were applied to simulate lateral and vertical movements of the camera due to car dynamics and road conditions. (17)
- **Brightness Adjustment**: To account for different lighting conditions, the brightness of the images was randomly altered within 20% to 120% of the original brightness, preparing the model for changes in time of day and weather conditions. (18)
- **Horizontal Flip**: To simulate driving scenarios in opposite directions or on different sides of the road, images along with their corresponding steering angles were flipped horizontally, with the steering angle inverted to maintain consistency. (19)

The implementation of these techniques was encapsulated in a function random_augment, which applies these transformations with a 50% probability for each image processed during training. This stochastic approach ensures a diverse set of images for each training batch, reducing the model's tendency to overfit to specific visual cues or scenarios.

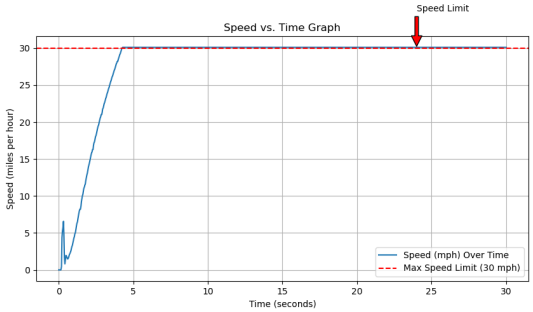

Figure 10: Performance on Track 1 with Data Augmentation

5.2.2 Training with Augmented Data

Utilizing a batch_generator, our training process continually fetched batches of augmented images. This generator dynamically applied the random augmentations only during the training phase, ensuring that the validation data remained unaugmented to provide an accurate assessment of model performance under standard conditions.

5.3 Performance Discussion

Figure 9 illustrates the training and validation loss for the improved model with data augmentation. The training loss for both models decreases rapidly, indicative of effective learning from the respective datasets. However, the key differentiator is observed in the validation loss. Unlike the baseline model, where validation loss plateaus early (as seen in 8), the improved model's validation loss continues to decrease, though at a slower rate, and stabilizes at a lower value. This suggests enhanced generalization capabilities, likely due to the diversified training scenarios introduced through augmentation.

Figure 10 and 21 show the vehicle achieving and maintaining the speed limit more consistently and managing obstacles with smoother decelerations and longer time till complete stop. This improvement is particularly evident in the vehicle's ability to approach and navigate through challenging scenarios with greater assurance and less intervention.

The analysis of the baseline and improved models illustrates the significant benefits of incorporating data augmentation in training autonomous driving systems. By introducing variability in training data, the improved model is better equipped to handle the complexities of real-world driving, resulting in a more robust, adaptable, and safe autonomous driving agent.

6 Further Exploration - Highway Gym Environment

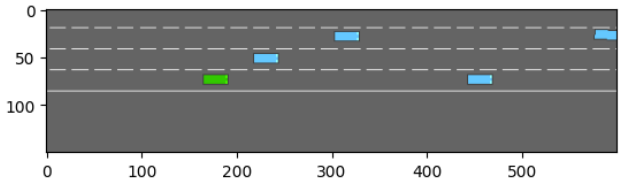

Figure 11: Highway Base Environment

6.1 Methodlogy

Three reinforcement learning algorithms were employed to train autonomous driving agents: Deep Q-Network (DQN), Proximal Policy Optimization (PPO), and a convolutional variant of DQN (DQN-CNN). These agents were trained on a base configuration of the highway environment and subsequently evaluated in three distinct scenarios: the standard environment, an environment with a higher number of lanes, and an environment with increased vehicle density.

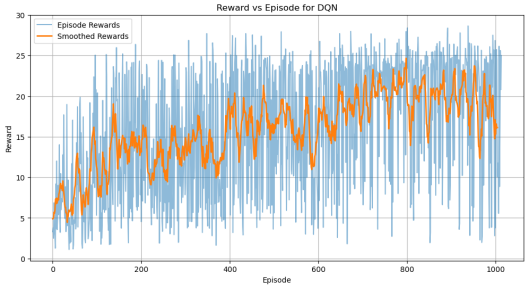

Figure 12: DQN Rewards over Episodes

6.2 Evaluation Metrics and Results

The agents' performances were quantitatively assessed based on the mean rewards obtained across multiple episodes. The results, depicted in 15, illustrate the comparative performance of each agent across different scenarios.

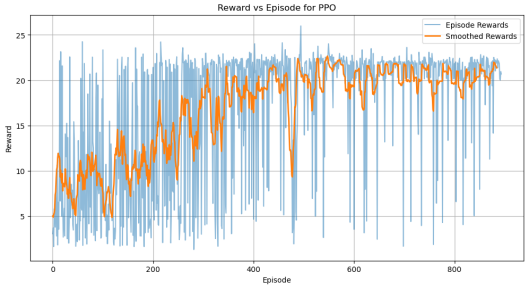

Figure 13: PPO Rewards over Episodes

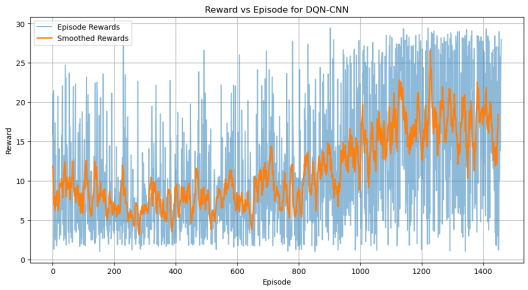

Figure 14: DQN-CNN Rewards over Episodes

Further insights were gained through the analysis of episode-by-episode reward patterns, as shown in Figures 12, 13, and 14 for DQN, PPO, and DQN-CNN respectively. These figures highlight the reward variability and the agents' learning progression over time.

6.3 Discussion

In the base environment, all agents performed comparably, demonstrating their capability to learn effective driving strategies under normal traffic conditions. However, distinctions emerged when the complexity of the environment was increased. In scenarios with a higher vehicle count, DQN-CNN demonstrated superior performance, likely due to its ability to process spatial hierarchies more effectively through convolutional layers, which is advantageous in densely populated traffic scenarios.

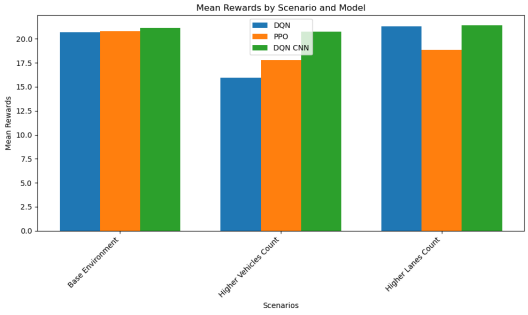

Figure 15: Evaluation Across Different Scenarios

References

[1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. "Backpropagation applied to handwritten zip code recognition." Neural Computation, 1(4):541–551, Winter 1989. URL: http://yann.lecun.org/exdb/publis/pdf/lecun-89e.pdf.

[2] M. Bojarski, D. Del 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." arXiv preprint arXiv:1604.07316, 2016. URL: https://arxiv.org/abs/1604.07316.

[3] D. A. Pomerleau. "ALVINN: An Autonomous Land Vehicle In a Neural Network." In Advances in Neural Information Processing Systems 1, NIPS Conference, 1989. URL: https://papers.nips.cc/paper/1989/file/8f4683d2b6c80226bf5e86a2e9ce26b3-Paper.pdf.

[4] S. Pan, Q. Yang. "A Survey on Transfer Learning." IEEE Transactions on Knowledge and Data Engineering, 22(10):1345-1359, October 2010. URL: https://ieeexplore.ieee.org/document/5288526.

Appendix

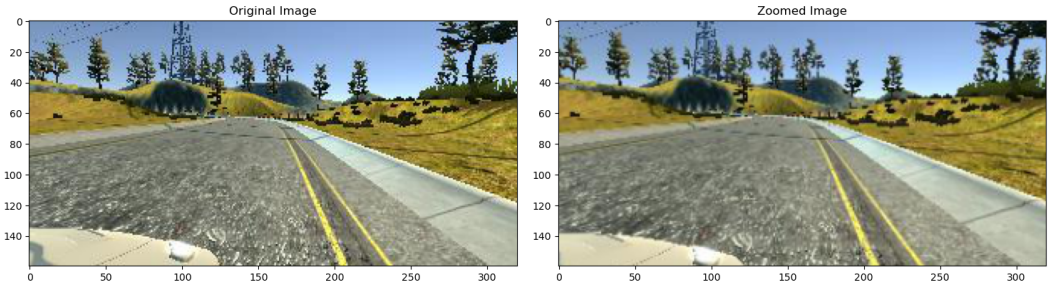

Figure 16: Zoomed Image

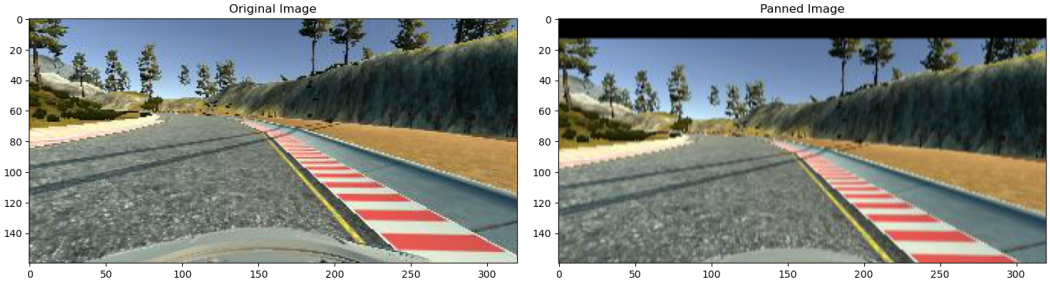

Figure 17: Panned Image

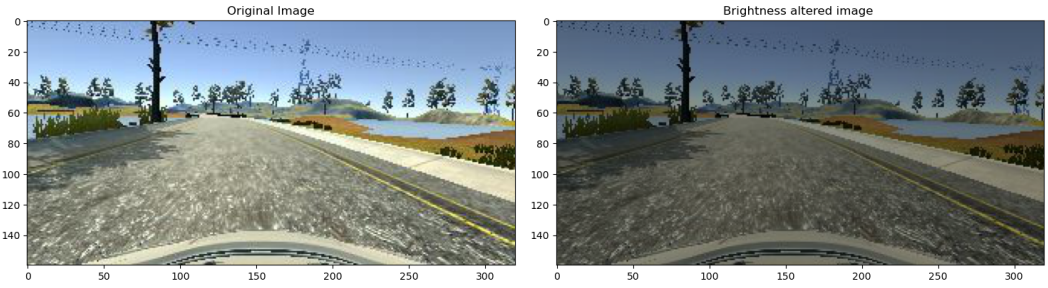

Figure 18: Bightness Altered Image

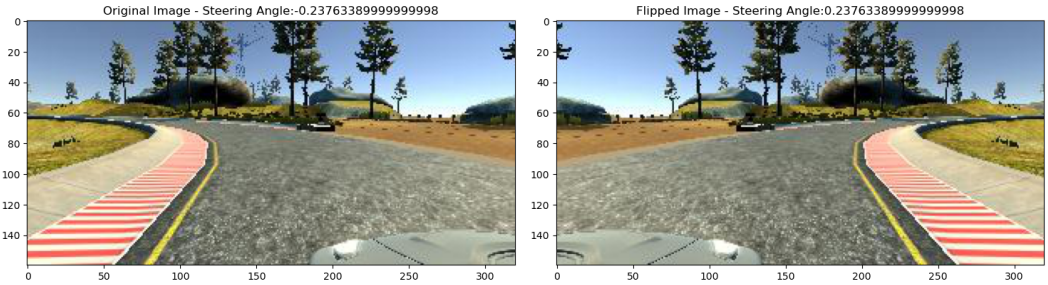

Figure 19: Flipped Image

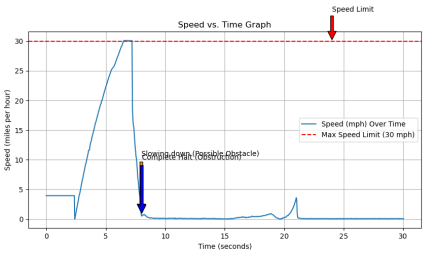

Figure 20: Performance on Track 2 without Data Augmentation

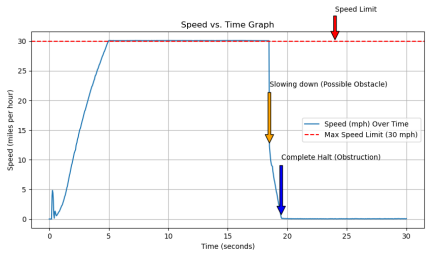

Figure 21: Performance on Track 2 with Data Augmentation