Self-Driving Car Simulation Project Using CNN

CVI620 Final Project, Winter 2025

Course Instructor: Ellie Azizi

1. Project Introduction

In this project, students will develop a neural network model to control a self-driving car. The goal is to determine the appropriate steering angle using images captured from the car's front camera to ensure the vehicle stays on the road.

The trained model will be tested by feeding it real-time images from the car's front camera, where it will predict the correct steering angle. To achieve this, we will use a simulation environment developed by Udacity and Nvidia.

2. Getting Started

It is recommended to create a virtual environment specifically for this project. (Your Computer Vision virtual environment is all good). A package_list.txt file is provided to ensure all necessary modules are correctly installed. This setup has been tested for TensorFlow GPU (64-bit) on Windows 10 to guarantee proper functionality.

Instructions for setting up the environment can be found in package_list.txt.

3. Data Collection and Creation

To collect the necessary data for the self-driving car model, download the self-driving car simulator from this link. If you are on MacOS or Linux you can search for "udacity self driving car simulator on mac". Extract the downloaded file. Locate and run the beta_simulator.exe file and a window similar to Figure 1 will appear. In order to set Up the Simulation adjust the settings as needed. Then, select "Play" to start the data collection process.

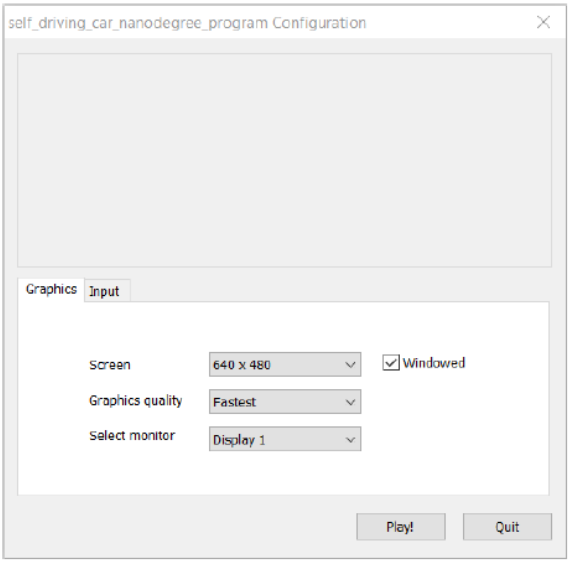

Figure 1.

After launching the simulator, the mode selection screen will appear.

- To collect data, select the "Training Mode" option.
- In this mode, you will manually drive the car along the correct path to generate and store the required data.
- This process ensures the model learns from properly labeled training data.

A visual representation of this screen is shown in Figure 2.

Figure 2.

Using the keys introduced in the Controls section, you can steer the car. After setting up the simulator for data collection, you must drive the car along a path (generally, the leftmost path is simpler and preferred for data collection).

This process should be performed in both directions to ensure balanced data collection.

To save the data, select the Recording option from the top menu, which is marked in red. This allows you to specify the path where you want to store the collected data (as shown in Figure 3).

Figure 3.

After selecting your desired path, click Select, and then start driving the car along the chosen route. When steering, it is better to use the mouse to maintain smooth and continuous movements. To collect a sufficient amount of data, drive about five times in the forward direction and five times in the reverse direction along the path.

After completing the car's movement along the entire path and clicking the Recording option, the simulator will begin saving the data, as shown in Figure 4. At the end of the process, a folder named IMG will be created in your project directory, containing all the recorded images from the car’s cameras. Additionally, a CSV file named driving_log.csv will be generated, storing all the movement data of the car along the path.

The columns in this file are organized as follows: Center, Left, Right, Steering, Throttle, Brake, Speed. The first three columns correspond to the images captured from the center, left, and right cameras. For this project, you will only use the center camera images and the Steering value, which represents the steering angle.

Figure 4.

4. Reviewing and Balancing the Dataset

To ensure that the car stays within the lane boundaries, the neural network must be trained in a way that prevents excessive deviation to the left or right.

To verify whether the collected data has a suitable distribution for training, a histogram should be plotted, specifically for the steering angle values. For a balanced dataset, the histogram should resemble the one shown in Figure 5.

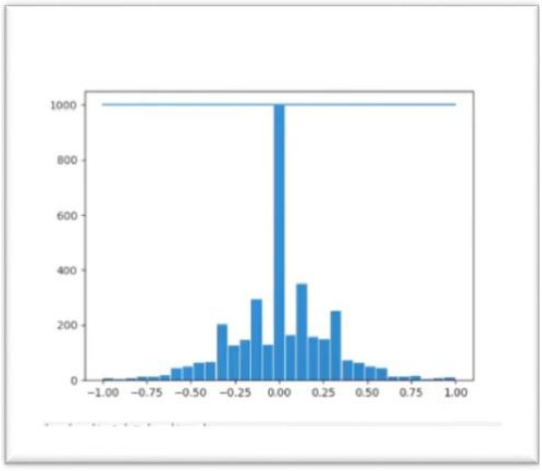

Figure 5.

4. Data Augmentation

To improve the generalization of the model, the collected images should be diverse. You can achieve this by applying data augmentation techniques such as: Flipping, Brightness adjustment, Zooming, Panning, Rotation. When applying flipping, make sure that the corresponding steering angle is also reversed (multiplied by -1). It is important that these transformations are applied randomly to only a portion of the data, rather than uniformly

across the entire dataset. Also, note that data augmentation should only be applied to the training dataset.

5. Data Preprocessing

The necessary preprocessing steps for this project include:

- Cropping the road area from the full image.
- Converting the image to YUV color space.
- Resizing the image to 200×66 pixels, as used by the Nvidia model.

The road area should be extracted for training, as shown in Figure 6.

Additional recommended preprocessing techniques:

- Normalization
- Applying a Gaussian Blur filter

Figure 6.

6. Batching the Dataset

Define a function that divides the dataset into batches of your desired size. This function will be used during training to efficiently process the data.

7. Training the Neural Network

At this stage:

- Build your neural network based on Figure 7.
- Plot and analyze the training graphs.
- Evaluate the training performance using these graphs to ensure proper learning.
- Save the final trained model for later use.

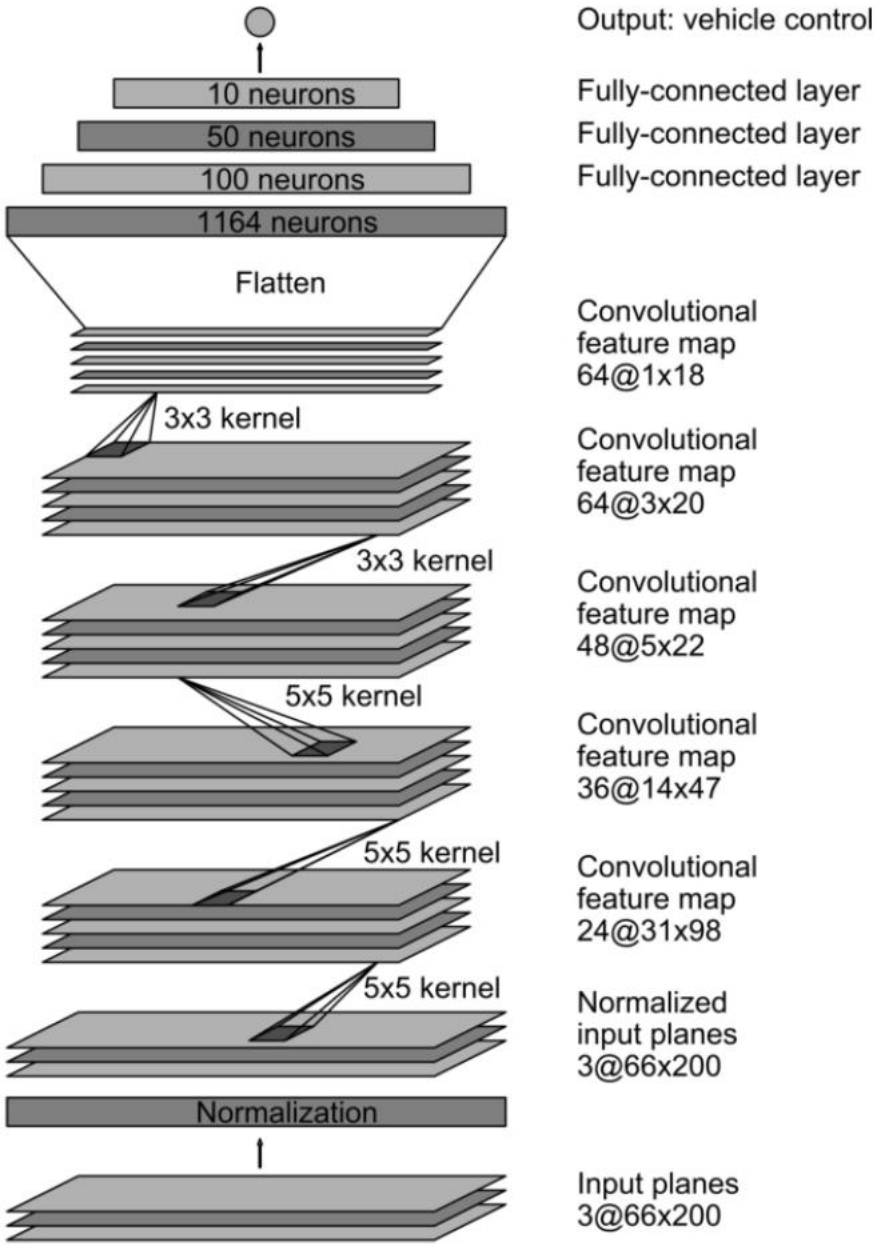

Figure 7.

8. Testing the Model

To use the final trained model in the simulator, you must run it within the same virtual environment that was set up initially.

This is necessary because some library versions may be incompatible, which could cause the simulator to malfunction.

Testing Steps:

1. Run the TestSimulation.py script.
2. Launch the simulator with the same settings used for data collection.
3. Observe the car's movement. It should follow the path accurately.

9. Deliverables

- All Python scripts, including: Data preprocessing, Training, Inference/testing, Data augmentation (if separate)
- A screen recording of the final trained model running successfully in the simulator
- Git repository, including: Clear commit history showing individual contributions, Proper project structure and organization
- Any documentation if needed like brief explanation of the approach or any challenges encountered and how they were solved, Instructions on how to run the project (e.g., environment setup, dependencies, how to launch training and testing scripts)

The project can be completed in groups of up to 3 individuals.

Good luck!