# AUTONOMOUS CAR USING DEEP LEARNING - FINAL REPORT

**Rudra Dey**
Student# 1010124866
rudra.dey@mail.utoronto.ca

**Pravin Kalaivannan**
Student# 1010141295
pravin.kalaivannan@mail.utoronto.ca

**Aadavan Vasudevan**
Student# 1010101514
aadavan.vasudevan@mail.utoronto.ca

**Abishan Baheerathan**
Student# 1010218756
abishan.baheerathan@mail.utoronto.ca

Total Pages: 9

## 1 INTRODUCTION

In 2023, the Toronto Police Service reported 298 vehicle collisions which led to death (or serious injury) within that year (Toronto Police Service, 2023). This data seems illogical when you find out that Toronto's rated safety score is 4.4 out of 5 compared to other cities in Ontario (BrokerLink Communications, 2025). This depressing reality of what a safe city is considered was our motivation to make this project about the creation of autonomous self-driving cars. To have self-driving cars be deployed and widely adopted in any urban city would significantly reduce traffic accidents.

In the short term, it is impossible to replace all human-driven cars with self-driven cars, instead a city needs to adopt the idea and slowly create infrastructure which supports these self-driving cars. Machine learning is a well-suited approach for this task because having a mix of self-driven cars and human-driven cars ultimately still involves human error, and thus requires on the spot judgement. A traditional rule based system will struggle to handle the wide variety of edge cases that occur on the road. Deep learning on the other hand enables the car to learn complex patterns from data and make rapid decisions. By using CNNs we can create a system which interprets multimodal inputs such as RGB camera sensors and indicator signal state, and then predict appropriate driving behaviour.

## 2 ILLUSTRATION

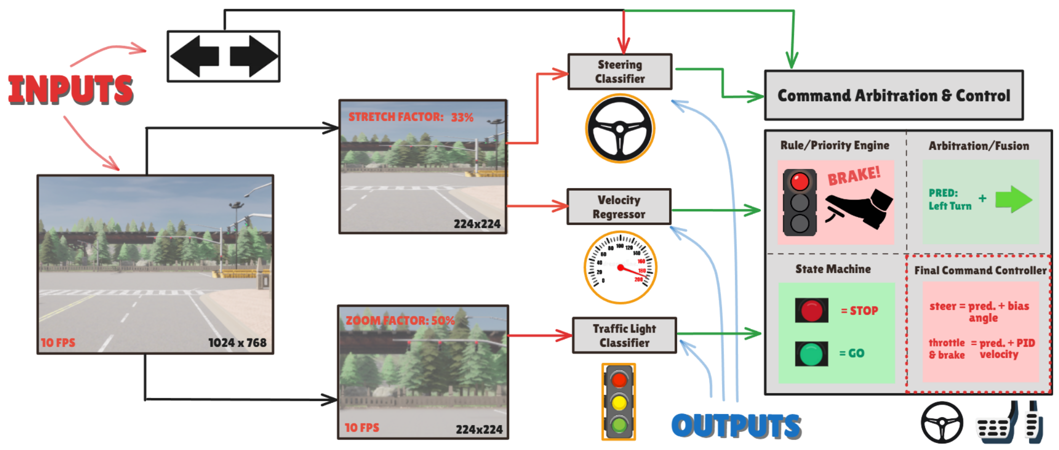


Figure 1: Overall illustration of the autonomous driving system.

## 3 Background & Related Work

Our autonomous self-driving car is a multimodal system, which predicts three outputs: steering angle, speed, and traffic light colour. This formulation puts our work in between two research fields in autonomous driving: raw-sensory input driving and control-oriented signalled driving. The following is an overview of five works related to these fields.

### 3.1 Vision-Based Multi-task Perception (Wang et al., 2025)

Recent studies have shown that multi-task learning can improve efficiency in autonomous driving by allowing a shared backbone to handle several perception and control tasks simultaneously. For example, models such as MultiNet and YOLOP perform object detection and lane segmentation, showing that the feature sharing reduces redundancy while maintaining real-time performance. Inspired by this approach, our project adopts a multi-head architecture that predicts steering, speed, and traffic-light state from a single camera input.

### 3.2 NVIDIA PilotNet Experiments (Bojarski et al., 2020)

NVIDIA's PilotNet by Bojarski et al. demonstrated that a CNN could map front-facing camera images directly to steering commands for lane detection and following. This showed that deep networks could replace hand-engineered pipelines for this task.

### 3.3 Conditional Imitation Learning (Codevilla et al., 2017)

Codevilla introduced Conditional Imitation Learning (CIL), which extended autonomous driving by conditioning the policy on higher-level navigation commands. The work showed that multi-task learning and structured outputs could be generalized in simulated urban driving. In our approach, we trained primarily via imitation learning, producing three outputs that were fed into a higher-level navigation module. This module applies conditional statements, which then control the car.

### 3.4 Traffic Light Detection (Pavlitska et al., 2023)

The team uses deep CNNs to classify signal states from camera crops. The problem of detecting and classifying traffic lights is a well-studied problem, and hard-engineered pipelines require specialized detectors and regional proposals. This showed that using a CNN a lightweight traffic-light classifier can be created, and that is what we did with high accuracy.

### 3.5 DeepDriving: Learning Affordance in Autonomous Driving (Chen et al., 2015)

DeepDriving is an approach for autonomous driving which predicts intermediate affordances, such as the distance to lane markings or the state of traffic lights, rather than controlling the car directly. We extended this idea and used models to predict steering angles, traffic-light states, and speed, which is then passed through a higher-level navigation command module, which then controls the car after conditions are met.

## 4 Data Processing

Data was collected from the CARLA simulator through the use of cameras and sensors, which makes up three different dataset types. We then normalized and resized the image as part of the preprocessing setup to implement the data into our network. Data augmentation techniques were then applied to force the traffic light model to learn deeper patterns.

### 4.1 Data Collection and Cleaning

This project required the team to gather five total datasets through the CARLA simulator: steering, velocity, traffic light, manual driving and finally a test dataset which included data from all four types of datasets. Four Python scripts were written to enable the process by which the data was collected. We attached RGB camera sensors to the car to record at 10 frames per second (FPS) as the car travels around designated maps, either through the built-in autopilot functionality or manual driving (for the manual driving dataset). In addition to the camera, sensors on the car collect necessary data for said dataset, such as velocity at that frame. The data collected from the sensors was used for ground truth labels, allowing the model to be trained. A single dataset is made up of various maps, which is done by saving an individual map's data as numpy files, then concatenating all the maps sequentially, creating one large dataset. Along with this data, our model needs live data collection for the steering signals. These turn signals are collected through a keyboard input, and sent to the model to bias the turning angle.

To ensure the data was not corrupted, scripts were created to visualize the images along with the associated ground truth labels. A numpy file viewer was also used to ensure the ground truth labels varied, meaning that the dataset was diverse.

Table 1: Datasets and their associated data types

| **Type of Dataset** | **Data Collected in a Frame** |
|---|---|
| Steering Dataset | Images, Angles, Turn Signals |
| Velocity Dataset | Images, Velocities |
| Intersection Light Dataset | Images, State Labels |
| Manual Driving Dataset | Images, Angles, Turn Signals, Velocities |

## 4.2 Data Preprocessing

All collected images were resized to a resolution of 224x224x3 as shown in Figure 2, with pixel values normalized to a range of [0, 1]. This was done to ensure that the images can be used with the ResNet18 backbone, allowing for use in our deep learning model. In addition to this, for the traffic light dataset, all images had the left and right sides blurred, along with a $\frac{1}{3}$ cropping of the bottom of the image, as shown in Figure 2. This was done to ensure the traffic light model learns the behaviour and patterns of the light ahead of it, while disregarding other signals' views.

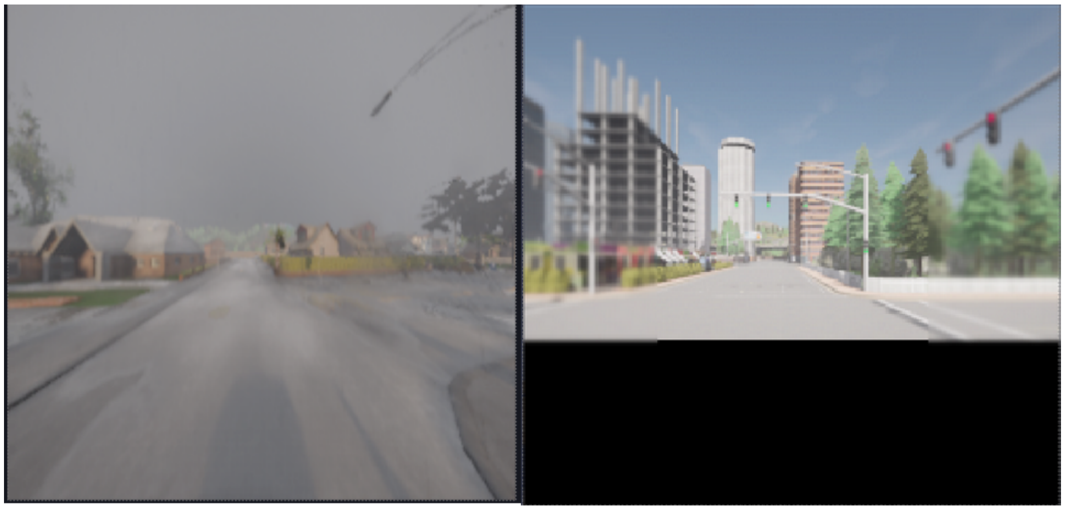


Figure 2: Preprocessed images: Steering/Velocity image (left), Intersection Light image (right).

## 4.3 Data Distribution

The datasets were split 80/20 for the training and validation sets, respectively, which ensured we have a large set of data to train the model, while having enough data for validating its performance. In addition, a separate test dataset was collected, containing a number of samples of the various types of data. This data was collected on an unused map (not seen in training or validation), allowing the team to test the performance of the model on unseen data. The test dataset also includes data from the manual dataset, which was split into its respective data types (steering, and velocity). All datasets were uploaded to a shared OneDrive so all members can access them as needed. Below, Table 2 shows the distribution of the data.

Table 2: Dataset Distribution

| **Dataset Type** | **Training Samples (80%)** | **Validation Samples (20%)** | **Test Samples** | **Total Samples** |
|---|---|---|---|---|
| Steering Dataset | 17,898 | 4,474 | 6,386 | 28,758 |
| Velocity Dataset | 56,000 | 14,000 | 8,000 | 78,000 |
| Intersection Light Dataset | 1,966 | 491 | 492 | 2,949 |
| **Total** | **75,864** | **18,965** | **14,878** | **109,707** |

## 4.4 Data Augmentation

The team iterated through different data augmentation techniques when training the models, and in the end decided to only augment the intersection light dataset for training.

For the steering and velocity models, the environmental context and surrounding structures are critical for how the model learns driving behaviours. If augmentations were applied to the images in the associated datasets, it would cause the model to pick up on the wrong patterns and learn bad driving behaviours, such as driving on the opposite side of the road if the augmentation was a horizontal flip.

In contrast, the intersection light dataset benefits from augmentations. A mix of augmentations was applied to ensure that the model learns the correct signal patterns. These augmentation techniques include brightness and contrast adjustment, horizontal flips, random rotation and cutouts. These augmentations, shown in Figure 3, allowed the model to learn deeper traffic light features, significantly improving its effectiveness.

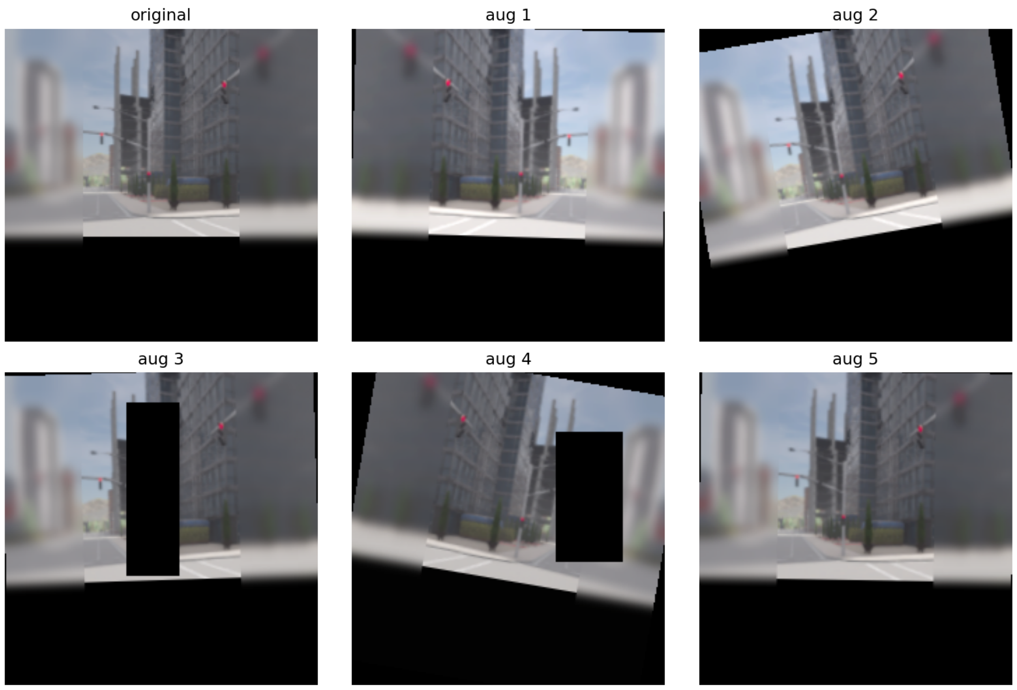


Figure 3: Example of Data Augmentation Techniques on Intersection Light Dataset

## 5 ARCHITECTURE

As shown in Figure 4 , our project uses three separate deep learning models, velocity prediction, traffic light classification, and steering angle prediction. All models are built on a shared convolutional backbone of ResNet-18 which is pretrained on ImageNet. ResNet was selected for its strong low-level and mid-level feature extraction ability, particularly in structured domains such as road images.

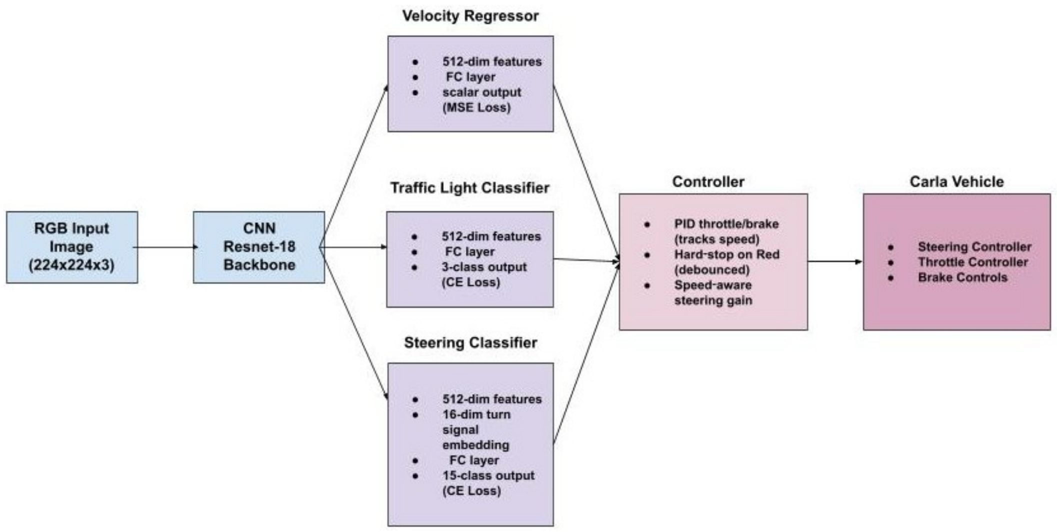


Figure 4: Final Architecture Low Level Diagram

### 5.1 TRANSFER LEARNING WITH RESNET-18

The ResNet-18 backbone processes each 224x224 dimensioned RGB frame to produce a 512- dimensional feature vector. For our application, only the convolutional layers of ResNet-18 are retained. In the early stages of training, these layers are frozen to preserve pretrained weights, with selective unfreezing applied during fine-tuning to adapt to the driving domain.

### 5.2 VELOCITY REGRESSOR

The velocity regressor receives the 512-dimensional feature vector and processes it through a two-layer fully connected network with a ReLU activation between layers. The first layer reduces dimensionality from 512 to 256 units, and the second produces a single scalar velocity value. Mean Squared Error (MSE) loss is applied during training to encourage accurate continuous speed prediction.

### 5.3 STEERING CLASSIFIER

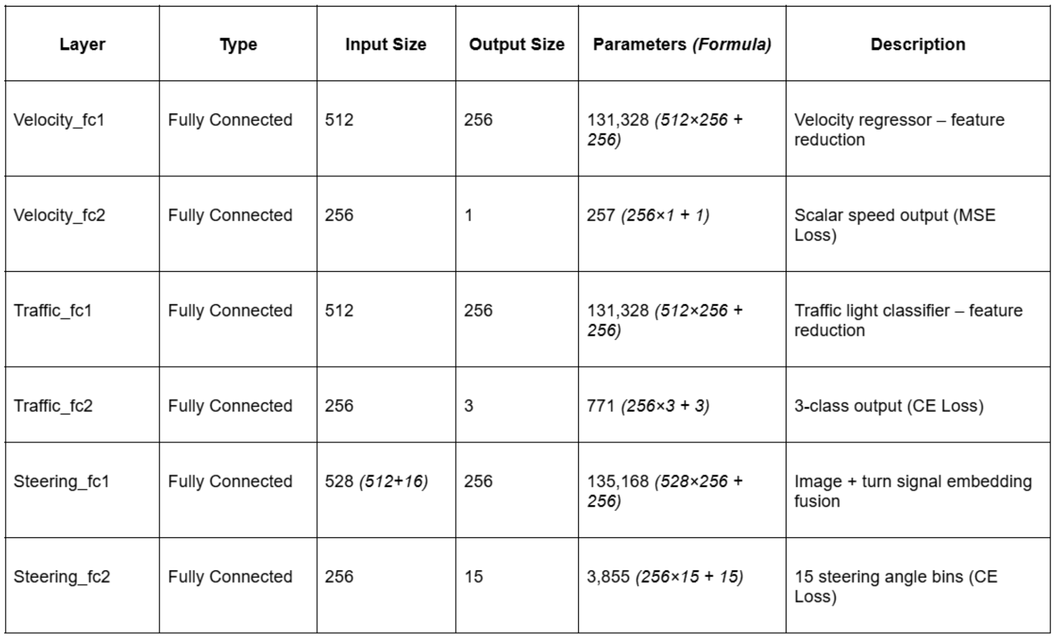
For steering control, the 512-dimensional backbone output is concatenated with a 16-dimensional learned turn-signal embedding, producing a 528-dimensional input vector. This vector passes through a fully connected layer reducing it to 256 units, followed by a ReLU activation, and finally a fully connected layer outputting logits for 15 discrete steering angle bins. Cross-Entropy loss is used for classification

### 5.4 TRAFFIC LIGHT CLASSIFIER

The traffic light classifier head reduces the 512-dimensional input to 256 via a fully connected layer with ReLU activation, followed by a final fully connected layer outputting probabilities for three traffic light states (Red, Green and No-Light). Cross-Entropy loss drives this classification task.

### 5.5 PARAMETER BREAKDOWN

Table 3 summarizes the custom fully connected layers in each prediction head, excluding the ResNet-18 backbone parameters ( 11M).



| Layer | Type | Input Size | Output Size | Parameters *(Formula)* | Description |
|---|---|---|---|---|---|
| Velocity_fc1 | Fully Connected | 512 | 256 | 131,328 *(512×256 + 256)* | Velocity regressor – feature reduction |
| Velocity_fc2 | Fully Connected | 256 | 1 | 257 *(256×1 + 1)* | Scalar speed output (MSE Loss) |
| Traffic_fc1 | Fully Connected | 512 | 256 | 131,328 *(512×256 + 256)* | Traffic light classifier – feature reduction |
| Traffic_fc2 | Fully Connected | 256 | 3 | 771 *(256×3 + 3)* | 3-class output (CE Loss) |
| Steering_fc1 | Fully Connected | 528 *(512+16)* | 256 | 135,168 *(528×256 + 256)* | Image + turn signal embedding fusion |
| Steering_fc2 | Fully Connected | 256 | 15 | 3,855 *(256×15 + 15)* | 15 steering angle bins (CE Loss) |

Table 3: Final Architecture Low Level Diagram

### 5.6 QUANTITATIVE RESULTS

### 5.7 QUALITATIVE RESULTS

## 6 BASELINE MODEL

Using a Ridge Regression Algorithm we created a baseline model that predicts steering angles for a self-driving car from grayscale camera images and turn signal inputs. This baseline model serves as a simple, interpretable baseline such that we can compare our more complex primary neural network model later.

### 6.1 RIDGE REGRESSION WITH IMAGE FEATURES

In the model outlined in Figure 5, the grayscale images (of shape 160x120) were flattened into 1D feature vectors and normalized. The features themselves represent the visual input of the car's front-facing camera. Additionally, left and right turning signals were captured as an additional feature; combining this with our flattened grayscale images we were left with a numpy array of shape $N \times 19201$, where $N$ represents the number of images, and 19201 are the number of feature column vectors. The performance was evaluated using Mean Squared Error (MSE) and $R^2$ Score on a held-out 20% test set.

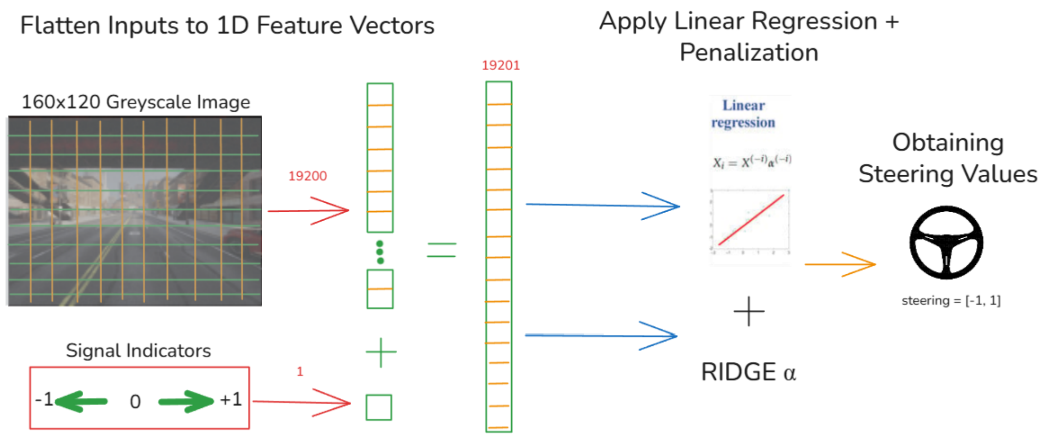

Flatten Inputs to 1D Feature Vectors
160x120 Greyscale Image
19200
19201
Signal Indicators
-1
0
+1
1
Apply Linear Regression + Penalization
Linear regression
RIDGE α
Obtaining Steering Values
steering = [-1, 1]

Figure 5: Basic Architecture of Steering Angle Ridge Regrsession Model

## 6.2 Minimal Tuning of $\alpha$

The only hyperparameter needing to be tuned was the regularization strength for the Ridge Regression Algorithm. This was manually tuned, and thus no validation set was used. A small set of candidate values for $\alpha$ was chosen: $\alpha \epsilon \{1, 10, 100, 250, 1000, 8000\}$ With each value of $\alpha$, the model was trained and tested with a dataset of 15 minutes of simulated driving. By obtaining the average MSE and the average $R^2$ score, our fourth model of $\alpha = 100$ came out to be the best.

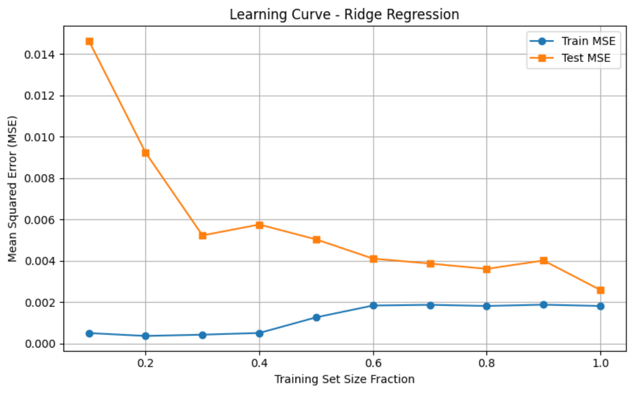


Figure 6: This is the learning curve for ”Model 4” a Ridge Regression Model with $\alpha = 100$

## 6.3 Overfitting and Model Quality

As shown in Figure 6, the results suggest some overfitting due to the model’s tendency to perform much better on training data than unseen data. In addition, as the training size increases the gap between Test and Train MSEs narrows, suggesting that with more data the model will generalize better.

## 6.4 Qualitative Observations

When running the steering angle prediction model in the CARLA simulator, the model exhibited promising behaviour in terms of autonomous driving capabilities, however with large limitations. The car consistently turned in the correct direction when prompted, it generally avoided obstacles but occasionally clipped walls. This suggests basic spatial awareness but limited precision. The car had trouble staying within lane boundaries, often drifting, especially during turns or on complex roads. Overall, despite limitations, the car performed reasonably well across different map layouts, showing decent control over turns.

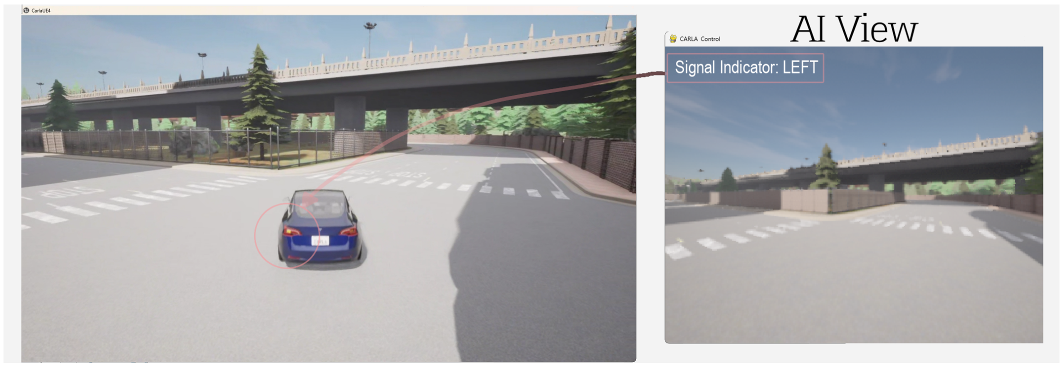


Figure 7: This is the demo for ”Model 4” in CARLA

## 7 Evaluation on Test Data

A test dataset was collected, made up of all the different dataset types (steering, velocity and traffic lights). This dataset contains data from an unused map, ensuring that it is unseen data. The images were then normalized and resized to 224x244x3in the same preprocessing as the training and validation data.

### 7.1 Test Results

Individual tests were run on the models to determine their performance, and the results are shown below:

- **Steering Classifier:**
  - Test Accuracy: 53.10%
  - Cross Entropy (CE) Loss: 1.792
- **Intersection Light Classifier**
  - Test Accuracy: 99.80%
  - CE Loss: 0.0457
- **Velocity Regressor:**
  - Mean Absolute Error (MAE) = 0.9614
  - Mean Square Error (MSE) = 1.7538
  - $R^2$ = 85.24%

### 7.2 Results Evaluation

For the Steering Classifier, the test resulted in an accuracy of 53.10%, which is lower than our expectations, but is reasonable given the complexity of the problem. As we output 15 classes, slight deviations in steering values can cause the predictions to fall into adjacent bins, lowering accuracy. Additionally, the model is designed to take in human input for steering values through turn signals. These signals bias the turning, making the model turn more, and as we are unable to apply these signals during testing, it might lead to a lower test accuracy. With a CE of 1.792 across the 15 classes, it is clear that the model learned some meaningful patterns, though it is not perfect.

The Intersection Light Classifier ended up with excellent results with a near-perfect test accuracy along with a CE loss of 0.0457. This shows that the classifier has learned deep features and is reliable at detecting the state of traffic lights.

The results of the Velocity Regressor meet expectations, with an MAE of 0.9614, meaning that the predictions were on average 0.96 m/s off the actual value, which is within our expectations. With a low MSE of 1.7538, it shows that the errors are not large discrepancies. Additionally, the $R^2$ of 85.24% shows the model captures the majority of the variation in velocity, demonstrating strong performance.

## 8 Discussion of Results

The results that the team collected show that our models performed well given the specific tasks that they had to complete. The steering classifier achieved an accuracy of 53.10%, the intersection light classifier achieved 99.80%, and the velocity regressor had a $R^2$ of 0.8524. The steering classifier accuracy is still impressive because of how difficult it is to predict steering angles and the different driving conditions. These results show that our system works well together and has achieved great results.

We implemented multimodal data by combining visual and sensor data. This was interesting because of how the models improved significantly. The system was able to capture crucial information in certain driving situations because of the multimodal data. This improved the system's accuracy because it helped provide more context for the models.

An important lesson that the team learned was to balance regression and classification loss. If this were done incorrectly, then one model would have overpowered the other ones. By tuning the weights, we improved the model's learning and achieved better overall results.

Overall, the team believes this system performs well and is able to accomplish its main goal, which is autonomous driving. The lessons learned from this project not only improved the system but also taught us about multimodal design, optimization, and data handling.

## 9 Ethical Considerations

If humans are over reliant on this system, they could be unprepared in scenarios where the system fails. Autonomous driving is considered an imperfect system. It will make mistakes, but a human driver should know how to step in and correct them. Furthermore, someone could decide to modify the system to follow unsafe driving habits. This would be a misuse of the system and raises safety concerns.Lastly, there are limitations to our model because it was tested more on urban maps. So, it adapts to the roads and driving style of an urban area. This affects how the model works in rural areas because the roads do not follow the same pattern. Also, this limitation raises concerns for safety because it was not tested properly in different settings.

## 10 Project Difficulty / Quality

Our project was difficult because it required us to integrate 3 models to handle completely different tasks. The team used a steering classifier, an intersection light classifier, and a velocity regression model. Each model collected its own data which was used to optimize the system. The steering classifier relies on various conditions, such as the angle of the road when turning, unpredictable behaviour from generated traffic, and the environment. Even with all these conditions, our model achieved an accuracy of 53.10%. This is impressive considering how the input is always changing. Collecting images of the traffic light was difficult because it is a small object in the simulator. Furthermore, the team was able to implement a multimodal architecture. This makes the project significantly more difficult because now you have to make sure the 3 models work together. Visual and sensor data were combined to improve the performance of our system. Also, the team tested data preprocessing and augmentation to handle various driving environments. Overall, the team learned a lot from developing a multimodal architecture. The models perform well given the difficulty of the project.

## REFERENCES

Mariusz Bojarski, Chenyi Chen, Joyjit Daw, Alperen Degirmenci, Joya Deri, Bernhard Firner, Beat Flepp, Sachin Gogri, Jesse Hong, Lawrence D. Jackel, Zhenhua Jia, B. J. Lee, Bo Liu, Fei Liu, Urs Muller, Samuel Payne, Nischal Kota Nagendra Prasad, Artem Provodin, John Roach, Timur Rvachov, Neha Tadimeti, Jesper E. van Engelen, Haiguang Wen, Eric Yang, and Zongyi Yang. The NVIDIA pilotnet experiments. *CoRR*, abs/2010.08776, 2020. URL `https://arxiv.org/abs/2010.08776`.

BrokerLink Communications. Top 10 most dangerous and safest cities to drive in ontario in 2024, May 2025. URL `https://www.brokerlink.ca/blog/top-10-most-dangerous-and-safest-cities-to-drive-in-ontario-in-2024`. Accessed: 2025-08-13.

Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2722–2730, 2015. doi: 10.1109/ICCV.2015.312. URL `https://ieeexplore.ieee.org/document/7410669`.

Felipe Codevilla, Matthias Müller, Alexey Dosovitskiy, Antonio M. López, and Vladlen Koltun. End-to-end driving via conditional imitation learning. *CoRR*, abs/1710.02410, 2017. URL `http://arxiv.org/abs/1710.02410`.

Svetlana Pavlitska, Nico Lambing, Ashok Kumar Bangaru, and J. Marius Zöllner. Traffic light recognition using convolutional neural networks: A survey, 2023. URL `https://arxiv.org/abs/2309.02158`.

Toronto Police Service. Total ksi, 2023. URL `https://data.torontopolice.on.ca/pages/total-ksi`. Accessed: 2025-08-13.

Hai Wang, Jiayi Li, and Haoran Dong. A review of vision-based multi-task perception research methods for autonomous vehicles. *Sensors*, 25(8), 2025. ISSN 1424-8220. doi: 10.3390/s25082611. URL `https://www.mdpi.com/1424-8220/25/8/2611`.