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: 8

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

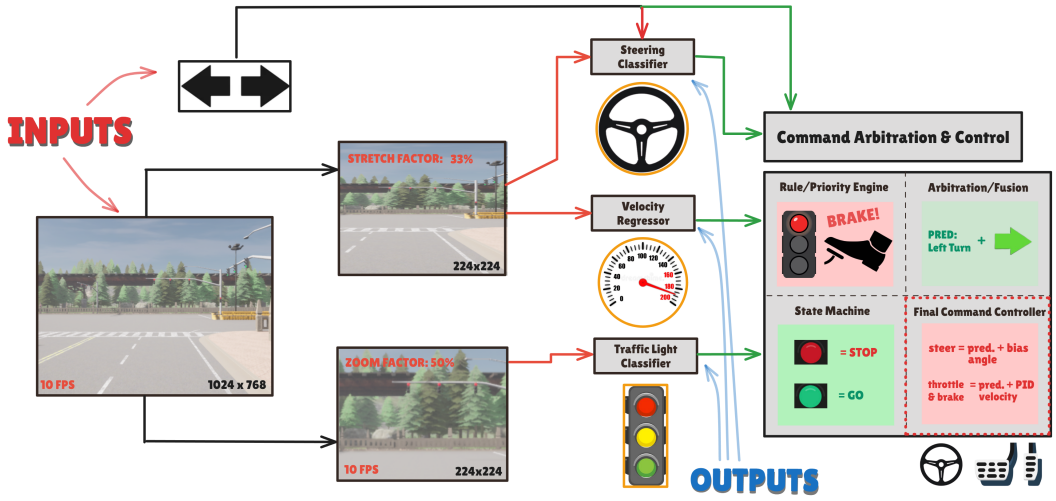

Figure 1: Overall illustration of the autonomous driving system.

3 BACKGROUND & RELATED WORK

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, manual driving, traffic light 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.

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 (Autopilot and Manual Driving)	Images, Angles, Turn Signals
Velocity Dataset	Images, Velocities
Intersection Light Dataset	Images, State Labels

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 examples on the left Steering/Velocity dataset image, and on the right Intersection Light image.

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. 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.

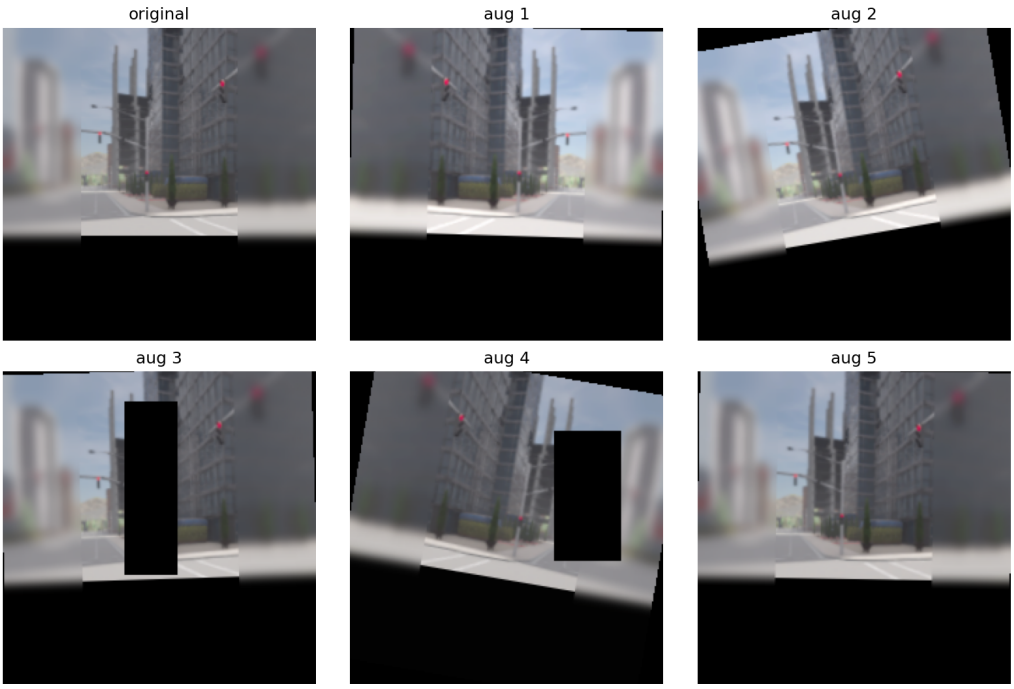

Figure 3: Example of Data Augmentation Techniques on Intersection Light Dataset

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.

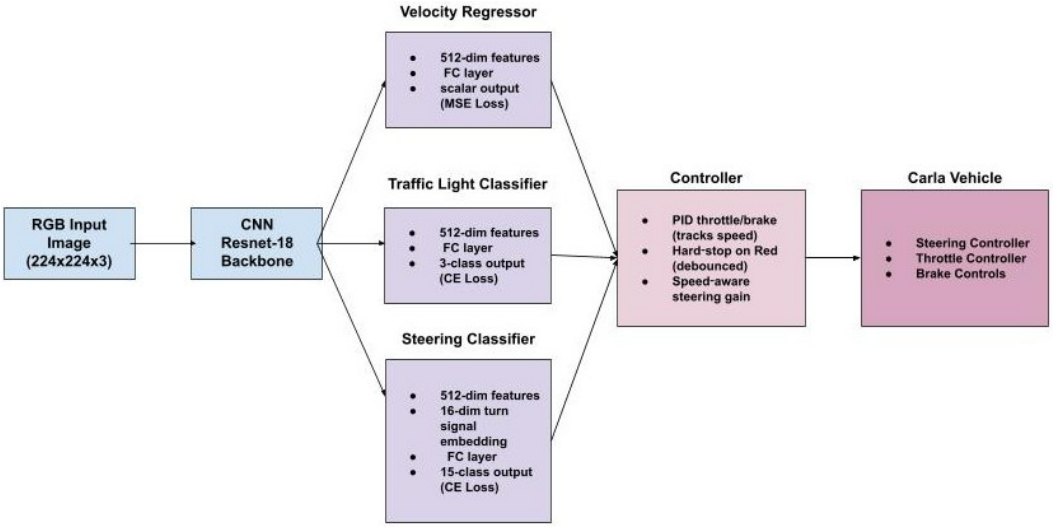

Figure 4: Final Architecture Low Level Diagram

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. To collect data, we used a simulator called CARLA, to obtain camera images, turn signal inputs, and our ground truth label for the steering angles.

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 Ridge Regression algorithm was selected because it penalized large coefficients (L_2 norm), allowing for a generalized model. A series of models were trained with different regularization strengths (α), and the performance was evaluated using Mean Squared Error (MSE) and R^2 Score on a held-out 20% test set.

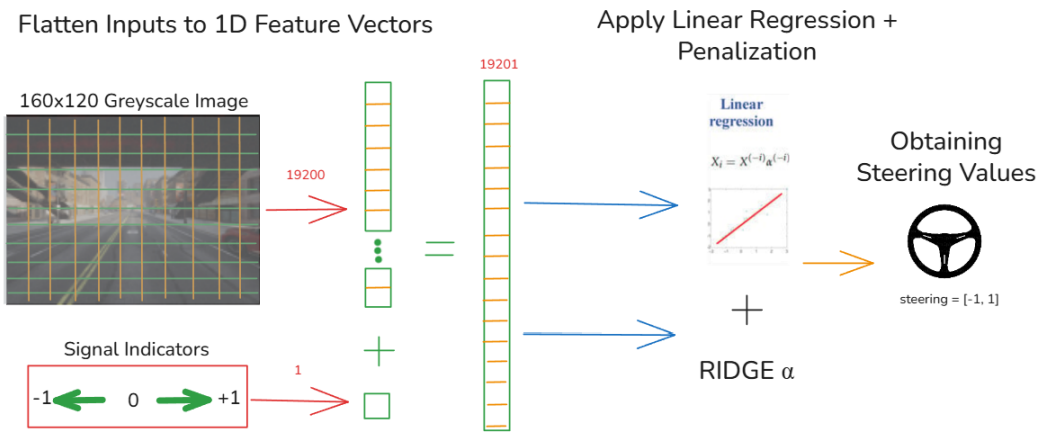

Figure 5: Basic Architecture of Steering Angle Ridge Regrsession Model

6.2 Minimal Tuning of α

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 α was chosen: $\alpha \epsilon \{1, 10, 100, 250, 1000, 8000\}$ With each value of α, 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.

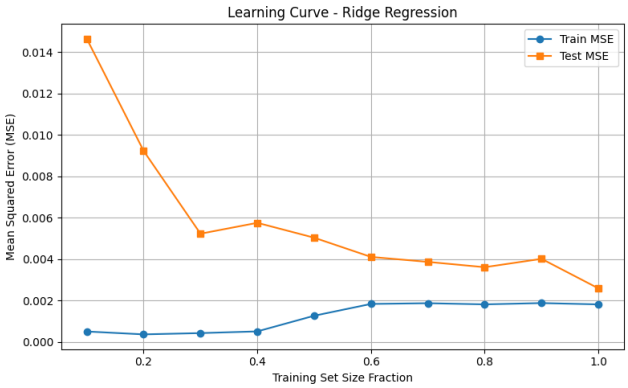

Figure 6: This is the learning curve for "Model 4" a Ridge Regression Model with $\alpha = 100$

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. This is reasonable since the training data frames, are all consecutive in nature, and the dataset used isn't extremely large either. 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. The final test MSE is (0.0025) which is quite low, meaning the model performs well overall after training on the data.

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 was able to drive freely, using the model's predictions from the front-facing grayscale cam-

era images and turn signals provided by the user. In Figure 7 we can see the model taking a left turn in an intersection.

- **Turn Signal Response:** The car consistently turned in the correct direction when prompted, showing that the model incorporated turn signal input effectively.
- **Obstacle Avoidance:** The vehicle generally avoided obstacles but occasionally clipped walls, suggesting basic spatial awareness but limited precision.
- **Lane-Keeping:** The car had trouble staying within lane boundaries, often drifting, especially during turns or on complex roads.
- **Map Generalization:** Despite limitations, the car performed reasonably well across different map layouts, showing decent control over turns.

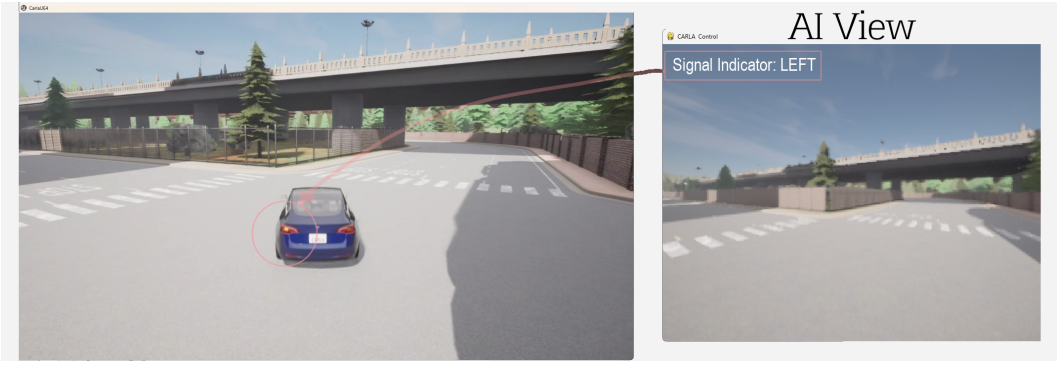

Figure 7: This is the demo for ”Model 4” in CARLA

7 Evaluate Model on New Data

8 Discussion of Results

9 Ethical Considerations

10 Project Difficulty / Quality

References

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.

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