AUTONOMOUS CAR USING DEEP LEARNING PROJECT PROPOSAL

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

ABSTRACT

This document presents our team's approach to developing an autonomous car system. This document contains research done on similar projects, data processing methods, architecture details, and a baseline comparison model. Our team looked into ethical concerns and potential major risks that could occur with this project. Further, the team came together to form a detailed project plan. This plan contains important tasks, who is responsible for each task, and project deadlines.
Total Pages: 8

1 INTRODUCTION

This project proposal will discuss our team's plan to train an autonomous car. The motivation behind this project comes from a shared interest in robotics and deep learning among the team. Our team's goal is to train the car to drive itself using cameras and control input. This project will be using a CNN model, public datasets such as the Udacity self-driving car dataset, and custom data that is collected through a mounted camera on an RC car. This is an interesting project because we get to explore the world of autonomous vehicles and the only sensor we plan on using is a camera. This project needs deep learning to process data in a fast and reliable way. We need to code procedural instructions for different situations based on our sensor (the camera), but this seems dangerous as humans can not think of every possible situation. A training model can handle this problem in a generalized way which is what we need to accomplish our goal.

2 ILLUSTRATION

3 BACKGROUND AND RELATED WORKS

3.1 AUTONOMOUS CAR RELATED PROJECTS

Through the use of deep learning, many autonomous car projects have been developed and tested, showing the potential of this technology in real-world applications. The following is an overview of five works related to autonomous vehicles using deep learning.

3.1.1 DEEP CNN END-TO-END LEARNING FOR AUTONOMOUS RC CARS (BHUTTA, 2023)

In this study, end-to-end learning using deep convolutional neural networks (CNNs) was used to autonomously control an RC car around a race track. A front-facing camera is used to collect images

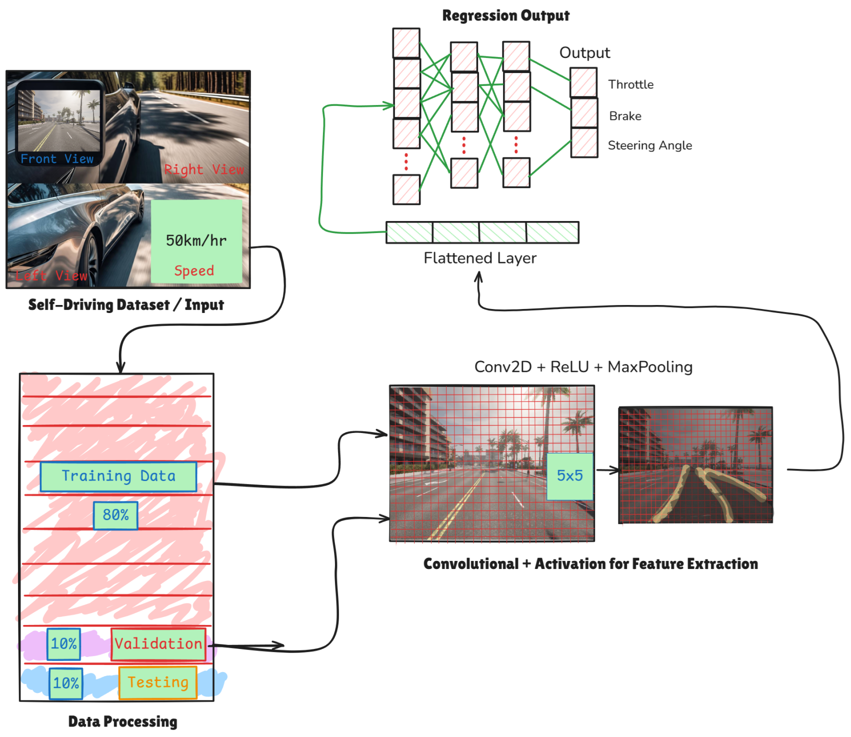

Figure 1: Illustration of self-driving car model

of the car's view, which is fed to the model. The CNN extracts features through convolutional and pooling layers, then applies the nonlinear ReLU activation function, which is then passed through 3 fully connected layers to generate the output of necessary steering angles and speed the car needs to travel (Bhutta, 2023).

3.1.2 Autonomous RC Car Using Neural Networks (Mallik, 2023)

This project explores autonomous driving through a small RC car using a Raspberry Pi and a CNN based end-to-end steering angle prediction system. To train the model, the car was manually driven around a track and the Raspberry Pi was used to track the analog steering inputs along with recording video frames taken at 10-20 fps. The project highlights limited computing resources from the Pi's onboard chips which may constrain CNN complexity along with data diversity being an issue where overfitting to a single environment leads to issues in other environments. Additionally, the project highlights the feasibility of deep learning for autonomous navigation and that it can be scaled up to larger robotic systems.

3.1.3 Development of Single-board Computer-based Self-Driving Car Model (Hamzah et al., 2022)

In this paper, an RC car based on the DonkeyCar project, powered by a Nvidia Jetson Nano and controlled by a CNN was used to drive autonomously around a track. The CNN uses live photos of the car's front view to predict the steering angle and speed required. The authors created varied training set sizes of 3000, 6000, 12000 and 24000 along with variations of network depth of between two and five layers of convolutional layers to find the best model (Hamzah et al., 2022). Their results showed that the model with 24000 images with three convolution layers performed the best with an absolute error at 0.18257 (Hamzah et al., 2022). Additionally, the results showed that having more data consistently reduced error, while adding layers above three had diminishing returns.

3.1.4 CNN based End to End Learning Steering Angle Prediction for Autonomous Electric Vehicles (Mygapula et al., 2021)

This paper explores CNN based end-to-end learning for steering angle prediction in autonomous vehicles using images captured of the front view of the car which is passed through to the model in a Jetson TX1. The team trained 3 different models based on different CNN architectures with different numbers of layers and the results showed that the CNN model with 4 CNN layers and 4 connected layers performed the best with 0.0354 test loss (Mygapula et al., 2021). These results highlight the viability of CNNs in learning steering behaviours compared to typical approaches in which systems are broken into multiple stages (such as road and object detection, and path planning)as it has fewer potential failure points.

3.1.5 End to End Learning for Self-Driving Cars (Bojarski et al., 2016)

In this paper, the team trained a CNN to map raw pixels from a front-facing camera directly to steering commands for a car. The network made up of nine layers, with one normalization layer, five convolutional layers and three fully connected layers was trained to minimize the mean squared error between steering commands and the human driver's inputs. Using only 72 hours of driving data, the team was able to successfully train the car to operate in diverse conditions showing how a CNN can perform the entire task without manual decomposition into smaller systems (Bojarski et al., 2016).

4 Data Processing

The Data Processing pipeline for the self-driving car project has three critical phases: Data Collection, Data Cleaning and Preprocessing, and Dataset Splitting. These phases are needed to ensure proper and high-quality training, validation, and testing of our model.

4.1 Data Collection

4.1.1 Public Datasets

- Udacity Self-Driving Car Dataset (Udacity, 2021)
- Berkeley DeepDrive (BDD100K) (Yu et al., 2018)

4.1.2 Custom Data Collection

- Simulated Data: Using CARLA Simulator to generate high-resolution images with corresponding control commands (Dosovitskiy et al., 2017)
- Real-World Data: Dash-mounted Camera on RC Cars.

4.2 Data Cleaning and Preprocessing

This phase involves preparing the image data and labels to maximize model accuracy. This includes image normalization and label consistency checks.

4.2.1 Remove Corrupted or Blurry Frames

Removal of incomplete metadata, heavy blur, overexposure, and distorted images using automated filters. These filters include: Laplacian Variance for blur detection, Histogram Analysis for overexposure, and Pixel Clipping Detection.

4.2.2 Image Resizing and Normalization

All images are resized to a uniform size of 320x240 pixels to match input expectations for deep CNNs. Additionally, pixel values are normalized to seed up convergence during training.

4.2.3 Label Alignment and Verification

Control labels are synchronized with their corresponding images using timestamps. Outliers and inconsistent data are flagged and removed. These control labels will be calculated with

4.3 Dataset Splitting

The dataset is split into:

- 80% Training
- 10% Validation
- 10% Testing

Since we are using data that comes from continuous streams, randomly shuffling data could have it so very similar frames of the stream can land in training, validation, and testing leading to memorization of the data. Thus, we will split the data chronologically, first 80% goes to training, the next 10% goes to validation, and the last 10% goes to testing. This way the test set includes unseen time segments from the drive.

5 Architecture

The neural network architecture is designed to take images of size 320x240x3 (height, width, RGB) of the cameras on the dash, left, and right side of the car, additionally, it will take in the speed of the car. With this, the neural network outputs/predictions are the continuous steering angle values for the car (regression task). Using multiple Conv2d layers, with 5x5 kernels, a stride of 1, and a ReLU activation to introduce non-linearity we aim to extract spatial features, like lane lines, edges, and road curvature. Using MaxPooling we shrink the image to reduce cost and overfitting, but we retain the features like lane edges or object boundaries. These feature maps are passed through a flattened layer and then into a fully connected layer which allows the model to learn high-level representations. The final output layer consists of three neurons with a linear activation function capturing the steering angle of the car, throttle, and brake. This implies that the speed of the car is constant from the start, or could be controlled manually.

5.1 Normalization

To improve generalization we:

- Batch normalization is applied after convolutional layers to normalize activations.
- L2 Weight Decay is applied during optimization to penalize overly complex models which leads to overfitting.

5.2 Training Method

This network is trained as a regression problem, where the goal is to minimize the error between predicted and actual steering angles. We will use the following to train our model:

- Loss Function: Using Mean Squared Error (MSE) to calculate the loss, penalizing large changes in steering predictions.
- Optimizer: Using the Adam Optimizer we will update network eights during training with adaptivative learning rates.
- Training Metrics: Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) are used to evaluate the accuracy of steering predictions.

6 Baseline Model

A hand-coded rule-based controller serves as a reasonable baseline for evaluating the performance of the autonomous car. This baseline model simulates traditional line-following logic, where decisions are made using simple heuristics based on pixel colour thresholds and pre-defined turning logic (Likmeta et al., 2020). The rule-based controller follows a deterministic flow, first, the incoming video frame is converted to grayscale or HSV colour space, and then a threshold is applied to detect the track line (usually a black line on a white surface or vice versa) (Likmeta et al., 2020). The centroid of the detected line is computed, and based on its position relative to the image center, turning decisions are made. For example, steering left if the line is on the left half of the image, right if on the right half, and forward if centered (Bojarski et al., 2016). This model does not involve learning or generalization; it is purely reactive and works well in constrained, consistent environments with high-contrast tracks. While simplistic, this approach is widely used as a baseline in autonomous driving projects due to its reproducibility and interpretability.

7 Ethical Considerations

7.1 Model Usage

Since this project involves physical hardware operating in real time, safety is a primary concern. The model may make poor decisions in edge cases (obstacles, unfamiliar lighting conditions), which could lead to hardware damage or unintended collisions (”davidad” Dalrymple et al., 2024). If used as part of a demonstration involving human interaction, ensuring safety measures (kill switches, limited speed) is essential. Another ethical concern is over-reliance on the model’s predictions. If used in future deployments (real vehicles or educational kits), assuming the trained model can handle all situations without proper validation may mislead users and cause harm or accidents (Laskey et al., 2017).

7.2 Data Collection

The data used for imitation learning is sourced from human demonstrations. A bias may arise if only a single driving style or track layout is captured. For instance, if the human driver consistently takes tight turns or drives aggressively, the model may learn to imitate that behaviour, limiting generalization to new tracks or drivers. Additionally, reinforcement learning episodes are generated in a simulated or controlled environment (Laskey et al., 2017). This might restrict the model’s robustness in diverse real-world settings. Ethical considerations also include ensuring no unnecessary wear is inflicted on hardware during data collection or that any modifications to the car setup are clearly documented and standardized.

8 Project Plan

To ensure the seamless progression of the project and collaboration among team members, we have devised guidelines and a timeline as seen below.

8.1 Team Guidelines

Meeting Guidelines	**Details**
In-person meeting	Location and time decided by the team
Online meetings	Tuesdays (3-5 pm) and Saturdays (1-3 pm)
Absence Notification	Must notify team 24 hours before the meeting if they will be absent

Table 1: Meeting Specifications

Collaboration Guidelines
• The team will be working together in a GitHub repository
• Each member is required to give a daily update message on their progress
• If conflicts arise, the team must discuss them together and decide on a solution that makes everyone happy
• Ensure the code has informational comments that help the team understand it

Communication Guidelines
• Group chat is active on weekdays and weekends (24/7)
• Group chat is active on weekdays and weekends (24/7)
• Members are required to frequently check discord messages (less than 2 hour response time)

8.2 Project Timeline

Task	**Assigned Team Member(s)**	**Description and Deadline**
Brainstorm Project Ideas	All team members	Discuss and generate ideas for project direction and RC car training scope. **Deadline:** June 9, 2025
Model Selection	All team members	Research relevant CNN architectures and choose suitable models. **Deadline:** June 10, 2025
Project Proposal (Draft)	All team members	Write first draft of proposal, including goals, methods, and background. **Deadline:** June 12, 2025
Project Proposal (Final)	All team members	Submit the final version of the proposal. **Deadline:** June 13, 2025
Data Collection & Setup	Rudra, Aadavan	Mount camera on RC car, gather video data, in addition gather simulator data. **Deadline:** June 20, 2025
Data Preprocessing	Rudra, Abishan	Resize, augment, and clean collected image data; prepare for model training. **Deadline:** June 27, 2025
Baseline Model Implementation	Aadavan	Build and test rule-based model for baseline performance comparison. **Deadline:** June 30, 2025
Architecture & Model Building	Pravin, Aadavan	Code CNN architecture, determine input/output layers and parameters. **Deadline:** July 4, 2025
Training Regime Development	Abishan, Pravin	Choose optimizer, loss function, learning rate; define training parameters. **Deadline:** July 8, 2025
Model Training & Tuning	Abishan, Rudra	Run training, adjust for overfitting, analyze learning curves. **Deadline:** July 10, 2025
Progress Report	All team members	Summarize work done so far and challenges, prepare a report for submission. **Deadline:** July 11, 2025
Model Evaluation & Testing	Aadavan, Pravin	Test trained model with live input, identify weaknesses, confusion matrix. **Deadline:** July 25, 2025
Final Report (Draft)	All team members	Write first full draft including abstract, results, ethics, background, etc. **Deadline:** August 1, 2025
Final Report (Final)	All team members	Submit the polished report with citations and final results. **Deadline:** August 8, 2025
Recording Presentation	Aadavan, Abishan	Record team sections, align visuals with speech. **Deadline:** August 12, 2025
Editing Presentation Video	Rudra, Pravin	Merge clips, finalize cuts, add transitions, captions, or diagrams. **Deadline:** August 13, 2025
Final Presentation	All team members	Present and submit final presentation. **Deadline:** August 15, 2025
Final Deliverable	All team members	Submit final code, data, report, and video via the course portal. **Deadline:** August 15, 2025

Table 2: Project Roadmap

9 Risk Register

The risk register contains some scenarios that could negatively impact our project. These potential risks negatively affect deadlines, quality of work, and more. Our team has discussed these scenarios and come up with solutions for each project risk.

Project Risk	**Likelihood**	**Solution**
A teammate drops the course	Unlikely	The team has to approach splitting up tasks differently as teammates will now have more responsibilities. We have to start tasks earlier and move internal deadlines to an earlier date because each member has more tasks to complete.
Model training takes longer than expected	Likely	It's common to procrastinate on less important or easier tasks, so this would be the time to complete them. The team should start working on tasks earlier, so if this does occur it doesn't affect project deadlines.
Experience hardware problems during testing	Likely	The team should keep extra hardware components in case they fail the during testing stages. Also, try not to overuse the physical components, and use simulation software to test.
The model works in simulation but not on a physical RC car	Likely	We shouldn't fully rely on simulations and should test the model on the physical RC car frequently. The earlier the team faces these issues, the longer we have to fix them. Compare the results from simulation and physical RC car and that might help solve problems.
If teammate misses an internal deadline	Likely	The team should hold everyone accountable as each member is responsible for their assigned tasks. The teammate should finish tasks as soon as possible after the missed deadline because this will hold back the team.

Table 3: Project Risks

10 GitHub Repository

`https://github.com/deyrudra/TrainingAutonomousCar`

References

Muhammad Raheel Bhutta. Deep CNN End-to-End Learning for Autonomous RC Cars. *Advances in Robotics and Mechanical Engineering*, 4:545–553, 2023. URL https://lupinepublishers.com/robotics-mechanical-engineering-journal/pdf/ARME.MS.ID.000181.pdf.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars, 2016. URL https://arxiv.org/abs/1604.07316.

David ”davidad” Dalrymple, Joar Skalse, Yoshua Bengio, Stuart Russell, Max Tegmark, Sanjit Seshia, Steve Omohundro, Christian Szegedy, Ben Goldhaber, Nora Ammann, Alessandro Abate, Joe Halpern, Clark Barrett, Ding Zhao, Tan Zhi-Xuan, Jeannette Wing, and Joshua Tenenbaum. Towards guaranteed safe ai: A framework for ensuring robust and reliable ai systems, 2024. URL https://arxiv.org/abs/2405.06624.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017. URL https://carla.org/.

Muhammad Shafly Hamzah, Nurfauzi Fadillah, Dwindra W. Maulana, I Made Joni, Camellia Panatarani, and Ferry Faizal. Development of single-board computer-based self-driving car model using cnn-controlled rc car. In *2022 International Conference on Electronics and Renewable Systems (ICEARS)*, pp. 1805–1812, 2022. doi: 10.1109/ICEARS53579.2022.9751873. URL https://ieeexplore.ieee.org/document/9751873.

Michael Laskey, Caleb Chuck, Jonathan Lee, Jeffrey Mahler, Sanjay Krishnan, Kevin Jamieson, Anca Dragan, and Ken Goldberg. Comparing human-centric and robot-centric sampling for robot deep learning from demonstrations, 2017. URL https://arxiv.org/abs/1610.00850.

Amarildo Likmeta, Alberto Maria Metelli, Andrea Tirinzoni, Riccardo Giol, Marcello Restelli, and Danilo Romano. Combining reinforcement learning with rule-based controllers for transparent and general decision-making in autonomous driving. *Robotics and Autonomous Systems*, 131:103568, 2020. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2020.103568. URL https://www.sciencedirect.com/science/article/pii/S0921889020304085.

Zaid Mallik. *Behind the Scenes of Our Self-Driving RC Car Project*. KaizenDev, 2023. URL https://www.kaizendev.tech/post/behind-the-scenes-of-our-self-driving-rc-car-project.

Prasad Mygapula, Adarsh Sasidharan, Sajith Variyar, and Soman Kp. Cnn based end to end learning steering angle prediction for autonomous electric vehicle. In *2021 Fourth International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pp. 2–7, 2021. doi: 10.1109/ICECCT52121.2021.9616875. URL https://www.researchgate.net/publication/356627453_CNN_based_End_to_End_Learning_Steering_Angle_Prediction_for_Autonomous_Electric_Vehicle.

Udacity. Udacity self-driving car dataset, 2021. URL https://github.com/udacity/self-driving-car. Includes driving data, annotated datasets, and open source code. Released under MIT and GPLv3 licenses.

Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A large-scale diverse driving video database, 2018. URL https://bdd-data.berkeley.edu. Released by Berkeley DeepDrive and BAIR. Dataset includes 100K driving videos with rich annotations such as object detection, drivable areas, lane markings, and instance segmentation.