Autonomous Kart Racing: End-to-End Self-Driving with Behavioral Cloning in CARLA Simulator

Francisco Ramos

***Abstract*— This paper presents a study on implementing behavioral cloning algorithms for autonomous driving in the CARLA simulator environment. Leveraging convolutional neural networks (CNNs), our research focuses on training models to predict steering angles based on input images, enabling autonomous navigation of racetracks. We explore two neural network architectures, SimpleNet and Net, assessing their performance in capturing driving behaviors and handling diverse scenarios. The results demonstrate promising accuracy and generalization, with the more complex Net architecture outperforming SimpleNet. Additionally, we augment the dataset with multiple camera perspectives to enhance model robustness and versatility. While our study highlights the potential of behavioral cloning for autonomous driving, further research is needed to address challenges such as real-world applicability and adaptation to varied driving conditions. Overall, our work contributes to advancing autonomous vehicle technologies, with implications for both recreational racing events and broader transportation applications.**

I. INTRODUCTION

Autonomous vehicles have transformed from a futuristic vision into a tangible reality on today's roads. These vehicles navigate urban environments through the integration of three core modules: perception, planning, and control, which collectively form a robust autonomous driving platform.

The adoption of autonomous vehicles offers several potential benefits, including enhanced safety, reduced driver workload, improved environmental efficiency, and increased comfort. Competitions such as the Autonomous Racing Karting series have further propelled advancements in autonomous vehicle technologies by challenging and innovating the capabilities of these systems.

Our project is affiliated with the Robot Open Autonomous Racing (ROAR) team. We leverage the CARLA simulator [5], which is configured to mirror the parameters of our actual system, allowing for rigorous software testing and development before deployment on real-world hardware.

The primary objective of this project is to implement a behavioral cloning algorithm. This algorithm trains an agent to emulate expert driving behaviors, enabling the vehicle to autonomously navigate a racetrack. Although behavioral cloning may not be the most scalable solution for autonomous driving—due to challenges such as scenario variability and trajectory inconsistencies—it remains invaluable. In racing contexts, particularly under conditions where primary algorithms fail or sensors malfunction (such as GPS failure under bridges), behavioral cloning provides a reliable fallback mechanism.

Furthermore, the specific nature of racetrack environments simplifies data extraction, making behavioral cloning an attractive approach for closed-circuit racing scenarios. We will discuss the strengths and limitations of this approach, including its adaptability to varied racing conditions and its potential as a supplementary strategy in autonomous vehicle technology.

II. BACKGROUND

The advent of convolutional neural networks (CNNs) has markedly advanced the field of pattern recognition. Traditional methods relied heavily on handcrafted feature extraction followed by classification. In contrast, CNNs automatically learn features directly from training data, a method particularly effective in image recognition tasks due to the convolution operation's natural fit for processing 2D images. This shift towards learned features has seen widespread adoption, bolstered by the availability of large labeled datasets like the Large Scale Visual Recognition Challenge (ILSVRC) and accelerated by the computational power of GPUs, which has dramatically enhanced both learning and inference speeds [1], [2], [3], [4].

In a seminal application of CNNs, NVIDIA demonstrated the feasibility of using a single front-facing camera to directly map raw pixels to steering commands in autonomous vehicles. This end-to-end learning approach, outlined in [1], allows the vehicle to operate in complex environments like traffic on local roads, highways, parking lots, and even unpaved roads without explicit road feature detection training. Instead, the system learns to interpret crucial road features solely from the driver's steering angles, leading to more compact network architectures and potentially superior performance due to the optimization of internal components tailored specifically to driving tasks [1].

The success of CNNs in these applications, however, is not devoid of limitations. Scalability issues arise when dealing with variable scenarios and environmental conditions. Moreover, CNNs trained on specific driving behaviors tend to replicate only those maneuvers learned during training, showing a significant limitation in their ability to adapt to new trajectories or maneuvers not covered in the training set. This limitation becomes particularly evident in scenarios where an image might be associated with multiple potential steering responses, thereby constraining the flexibility of the trained model in real-world operations.

Our work is thus positioned within this ongoing exploration of autonomous vehicle technologies, seeking to refine and expand upon the neural network methodologies that have shown promising results but also exhibit clear areas

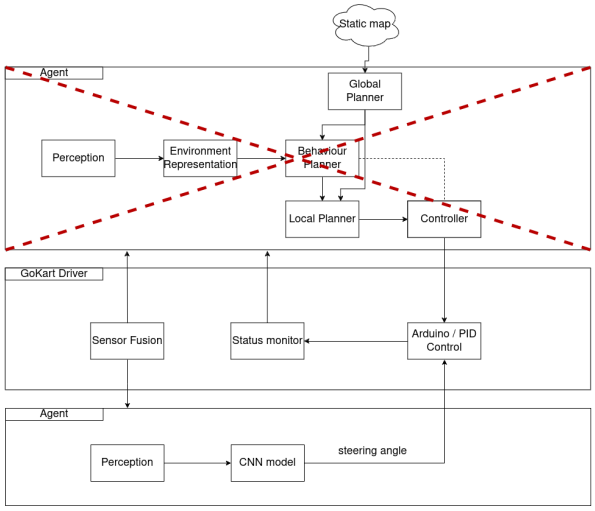


Fig. 1: Control diagram depicting the integration of CNN for real-time steering prediction.

for improvement in terms of versatility and adaptability to diverse driving conditions.

III. METHODOLOGY

This section outlines the methodology used for integrating convolutional neural networks (CNNs) into our autonomous GoKart project. The process involves data collection through simulation, preprocessing, model training, and real-time steering angle prediction.

A. Control Diagram

Our system replaces traditional sensors and planners with a CNN that processes images from a front-facing camera to predict steering angles directly. Figure 1 illustrates this integration. The CNN acts as the perception module, receiving raw image data and outputting predicted steering angles for the PID controller to execute.

B. Data Collection

The dataset essential for training our convolutional neural network (CNN) is procured using the CARLA simulator. To facilitate this, a dedicated node has been developed. This node actively monitors and captures data from two primary sources: the camera feed and vehicle state topics. The captured images, along with their corresponding steering angles, are stored in a designated folder. Each entry is cataloged in a JSON file that contains paths to the images and their associated labels.

For the data acquisition process, a human expert manually controls the vehicle within the simulation to execute specific driving maneuvers. This manual control ensures that the data reflects realistic driving patterns and helps introduce variability in the dataset. To further enhance the diversity of the training data, the same trajectory is driven multiple laps under varying conditions.

Additionally, to broaden the model's applicability to different driving scenarios, we are compiling multiple datasets, each representing unique trajectories and environmental conditions. This diversified dataset strategy aims to robustify the CNN's adaptability to changes in driving environments.

The control setup and the recording methodology are illustrated in Figure 2, which provides a visual representation of the data collection framework.

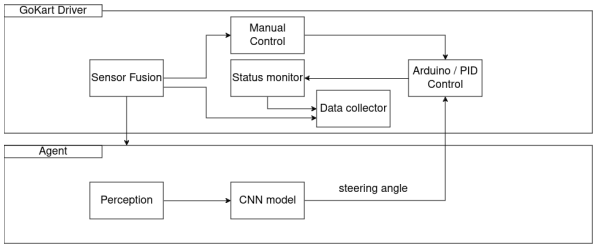


Fig. 2: Schematic of the data collection setup using the CARLA simulator, depicting the integration of manual control and data recording nodes.

C. Data Processing

The data processing pipeline initiates with label normalization where steering angles are rescaled to a range from 0 to 1, with 0 representing a full left turn and 1 a full right turn. This normalization is fundamental for training the CNN effectively.

Further, data loaders play a pivotal role in efficient data management during model training. They are employed to ensure that images are not only normalized but also resized to a consistent dimension of $256 \times 256 \times 3$, down from their original size of $376 \times 1344 \times 3$ (height, width, channels). This resizing might lead to a loss of some detailed features; however, such detail is extraneous as the model's primary objective is to discern the road's layout and generate corresponding steering commands.

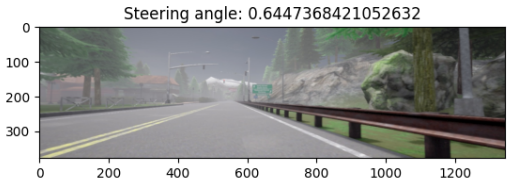
The standardization of images involves normalizing pixel values to a standard scale using predefined means and standard deviations (mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225]). This step is crucial for maintaining computational efficiency and consistency during training, and it's detailed in the project's code repository.

Examples of a raw and a transformed image are presented in Figure 3 to illustrate the preprocessing effects.

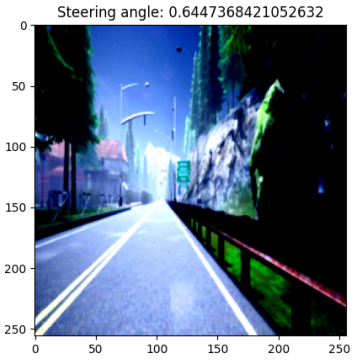
Concluding the preprocessing phase, the dataset is segmented into training and validation sets with an 80% split favoring training. This approach is tailored to mimic expert driving behavior within specific scenarios, rather than aiming for generalization across unseen environments. This method aligns with the project's objectives of behavior cloning on predetermined circuits.

D. Data Augmentation with additional cameras

Following a similar approach to that described in [1], we have enhanced the robustness and versatility of our model by expanding our data collection setup. Two additional cameras have been integrated, each positioned at ±30 degrees relative to the front-facing (0-degree) camera. This configuration



(a) Example of a Raw Image from the Dataset.



(b) Example of a Transformed Image.

Fig. 3: Comparison of a raw and transformed image to demonstrate the preprocessing effects.

broadens the range of captured perspectives, simulating more complex driving scenarios and enriching the training data available for our model.

Fig. 4: Sketch of data augmentation and perception field with additional cameras.

Each camera feeds into the same data collection system but with a crucial modification: the steering angles associated with the images from the left and right cameras are adjusted to reflect their respective vantage points. Specifically, a predetermined amount is added to the steering angles for images from the left camera, and an equivalent amount is subtracted from those captured by the right camera. This adjustment compensates for the angular displacement of the cameras and helps in training the model to correct its orientation when faced with similar skewed perspectives.

This augmented dataset is particularly valuable as it trains the vehicle to recognize and correct for improper orientations. When the vehicle encounters a situation where it is not optimally aligned with the road, it can refer to these trained responses to adjust its steering angle accordingly and return to the correct orientation.

E. Models (Architectures)

a) Neural Network Architectures: Two neural network models, SimpleNet and Net, are developed to explore the trade-offs between simplicity and complexity in the context of behavioral cloning for autonomous driving. SimpleNet adopts a straightforward architecture with fewer layers, whereas Net incorporates a more complex design inspired by Nvidia's architecture for self-driving cars [2].

SimpleNet consists of two convolutional layers with 24 and 36 filters, both using 5x5 kernels and a stride of 2. This is followed by three fully connected layers with sizes 100, 50, and 1 unit respectively, with ReLU activations and a dropout rate of 0.1 for regularization.

Net expands on this by including five convolutional layers with varying numbers of filters and kernel sizes, alongside more complex fully connected layers that progressively reduce from 100 to 10 units before reaching the output. It also uses ReLU and dropout at a rate of 0.1.

Both models are tailored to process input images and predict steering angles, with the difference in architecture intended to assess the impact of network complexity on the accuracy and efficiency of behavioral cloning in real-world driving scenarios. A summary of these architectures is shown in Table I.

TABLE I: Comparison of Neural Network Configurations

Config	SimpleNet	Net
Input Channels	3	3
Convolutional Layers	2 (24 filters, 36 filters)	5 (24, 36, 48, 64, 64 filters)
Conv Kernel Sizes	5x5	5x5, 3x3, 3x3, 3x3, 3x3
Strides	2	2, 2, 2, 1, 1
Fully Connected Layers	3 (100, 50, 1 units)	4 (100, 50, 10, 1 units)
Dropout	0.1	0.1

IV. IMPLEMENTATION

A. Model Training

Both models were trained using a backpropagation algorithm with Adam optimizer, chosen for its effectiveness in handling sparse gradients and its adaptive learning rate capabilities. Training was conducted over 50 epochs with a batch size of 32. The loss function used was Mean Squared Error (MSE), which is standard for regression tasks such as predicting steering angles. Early stopping was implemented to prevent overfitting, ceasing training if validation loss did not improve for 5 consecutive epochs.

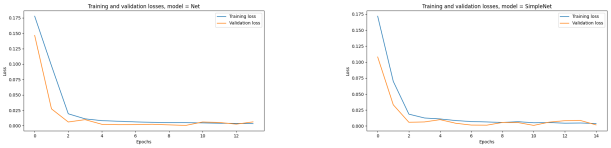
B. Validation

Model validation was performed at the end of each training epoch to monitor performance and adjust hyperparameters accordingly. Validation metrics included MSE and Mean Absolute Error (MAE) to assess the accuracy of steering angle predictions. The validation set comprised 20% of the total dataset, selected randomly to ensure diversity.

V. RESULTS

A. Train and Validation Results

During the training and validation phases, the performance metrics indicated that Net achieved lower Mean Squared Error (MSE) and Mean Absolute Error (MAE) than SimpleNet, demonstrating superior accuracy and generalization on unseen data. The analysis of the loss curves revealed a consistent reduction in training loss, although a plateau towards the later epochs suggests that extending training duration may not result in significant further improvements.



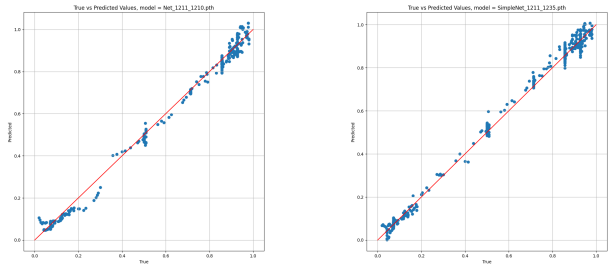
(a) Net (b) SimpleNet

Fig. 5: Comparison of Training and Validation Loss Curves for Neural Network Models

TABLE II: Validation Results for the SimpleNet and Net Models

Model	Mean Absolute Error (MAE)
SimpleNet	0.0214
Net	0.0231

We further assessed model performance by comparing predictions against actual non-dimensional steering angle values on the validation dataset. Figure 6 displays a scatter plot of actual versus predicted values, where a line with a slope of 1 indicates perfect prediction accuracy.



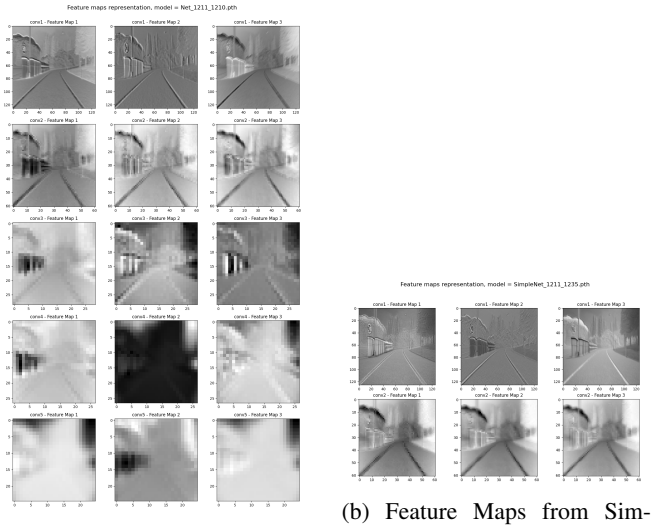
(a) Validation Performance of Net Model (b) Validation Performance of SimpleNet Model

Fig. 6: Validation Data Model Fitness Comparison

B. Feature Maps

To deepen our understanding of the convolutional layers' functionality within our models, we examine the feature maps produced by these layers. These visualizations help illustrate how the models process and interpret input images, particularly in the context of driving scenarios. Feature maps reveal the patterns and spatial hierarchies that the neural networks have learned, providing insight into the different features each model emphasizes during training.

Figure 7 shows the feature maps from selected convolutional layers of both models. Each subfigure represents a layer's output when processing the same input image, highlighting the distinct ways in which SimpleNet and Net extract and prioritize various features.



(a) Feature Maps from Net (b) Feature Maps from SimpleNet

Fig. 7: Visualization of Convolutional Layer Outputs for Net and SimpleNet Models

C. Inference on Images

To evaluate the effectiveness of the trained models, inferences were conducted on a sequence of images extracted from a video that represents a complete lap of the circuit used for data collection. This approach allows us to directly compare the predicted steering angles with the expert's actual maneuvers during the lap. Visual aids such as a steering wheel graphic and a comparison bar are incorporated into the images to enhance the visualization of the model's predictions.

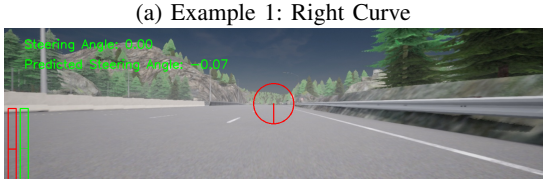
The inference results are displayed in Figure 8, showcasing how the models handle different driving scenarios like curves and straight paths. Further, Figure 9 presents a detailed comparison of predicted versus actual steering angles over a sequence of video frames, highlighting the models' performance over continuous driving sequences.

The mean absolute error (MAE) for each model is tabulated below, including results from applying a noise-reducing exponential filter to the output.

TABLE III: Mean Absolute Errors for Different Models

Model	Mean Absolute Error (degrees)
SimpleNet	0.3037
SimpleNet (Filtered)	0.2081
Net	0.3181
Net (Filtered)	0.2052

(a) Example 1: Right Curve



(b) Example 2: Straight

(c) Example 3: Left Curve

Fig. 8: Inference Examples Illustrating Different Driving Scenarios (Steering angles are in degrees).

For a live demonstration of the video inference, follow this link: `https://drive.google.com/file/d/1FAz_LlDRw2X5yyNE_WU86OxmwSWP06J4/view?usp=sharing`.

D. Inference on Real-Time Control

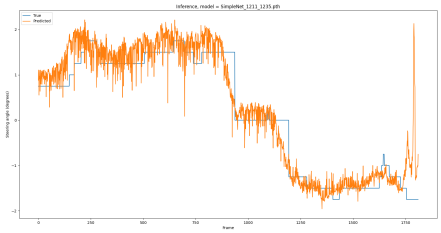
The models were further tested in a dynamic environment using the CARLA Simulator to demonstrate real-time control capabilities. This setup enabled the models to autonomously navigate the simulated vehicle based on real-time image inputs, effectively showcasing their ability to make immediate driving decisions.

Experience this live demonstration by visiting the link: `https://drive.google.com/file/d/138B57XoGlEkzMtahvRXlAw-HaDrrmjFp/view?usp=drive_link`.

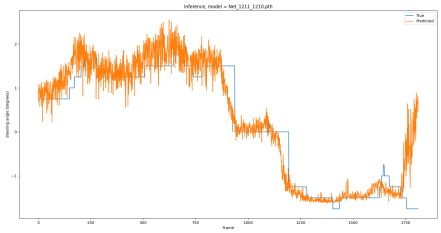
VI. DISCUSSION

The results of our study demonstrate the feasibility and effectiveness of implementing behavioral cloning algorithms for autonomous driving in the CARLA simulator environment. Through the integration of CNNs, we have successfully trained models to predict steering angles based on input images, enabling the simulated vehicle to autonomously navigate a racetrack.

One notable success of our approach is the achievement of low MAE during both training and validation phases. Our more complex model, Net, outperformed the simpler SimpleNet architecture, showcasing the importance of network complexity in capturing the nuances of driving behaviors. Additionally, the augmentation of the dataset with additional camera perspectives helped improve the robustness and versatility of our models, particularly in handling skewed viewpoints and diverse driving scenarios.



(a) Predicted vs. True Steering Angle Sequence for SimpleNet model for a selected track



(b) Predicted vs. True Steering Angle Sequence for Net model for a selected track

Fig. 9: Comparison of Predicted and True Steering Angle Sequences in a Video for Models with Different Complexities

However, our study also reveals several limitations and areas for improvement. Despite the overall success of our models, there were instances where predictions deviated significantly from the ground truth, especially during complex maneuvers such as sharp turns or sudden lane changes. This indicates the need for further refinement and optimization of the neural network architectures to better generalize across a wider range of driving scenarios.

Furthermore, while our models demonstrated promising performance in the simulated environment, their real-world applicability may be limited by factors such as variability in road conditions, unpredictable obstacles, and environmental factors like lighting and weather conditions. Future research should focus on enhancing the robustness and adaptability of behavioral cloning algorithms to ensure safe and reliable autonomous driving in diverse real-world settings.

VII. CONCLUSION

In conclusion, our study highlights the potential of behavioral cloning algorithms for autonomous driving applications, particularly in closed-circuit racing scenarios simulated in environments like CARLA. By leveraging convolutional neural networks, we have demonstrated the ability to train models that can accurately predict steering angles based on input images, enabling autonomous navigation of racetracks.

While our results are promising, there is still much room for improvement and further research. Enhancing the robustness and adaptability of our models to handle diverse driving scenarios and real-world conditions remains a key challenge. Additionally, exploring advanced techniques such as reinforcement learning and model fusion could lead to further advancements in autonomous driving technology.

Overall, our study contributes to the ongoing efforts to develop safe, efficient, and reliable autonomous vehicle systems, with potential applications ranging from recreational racing events to commercial transportation and beyond.

VIII. ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to all those who contributed to the success of this project. Special thanks to the members of the Robot Open Autonomous Racing (ROAR) team at the University of California, Berkeley for their support, technical expertise, and valuable insights throughout the development process. We would also like to extend our appreciation to the organizers and developers of the CARLA simulator for providing an invaluable platform for experimentation and research in autonomous driving.

REFERENCES

[1] M. Bojarski et al., "End to End Learning for Self-Driving Cars," arXiv preprint arXiv:1604.07316, 2016. [Online]. Available: https://arxiv.org/abs/1604.07316

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[3] L. D. Jackel, D. Sharman, Stenard C. E., Strom B. I., and D Zuckert, "Optical character recognition for self-service banking," *AT&T Technical Journal*, vol. 74, no. 1, pp. 16–24, 1995.

[4] "Large Scale Visual Recognition Challenge (ILSVRC)." [Online]. Available: http://www.image-net.org/challenges/LSVRC/

[5] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.