

A Review on DonkeyCar Autonomous Driving Platform Using Neural Network

Yangyang Li¹ and Jian Qu^{2*}

^{1,2}Faculty of Engineering and Technology, Panyapiwat Institute of Management, Nonthaburi, Thailand

E-mail: 6672100032@stu.pim.ac.th, jianqu@pim.ac.th*

Received: September 26, 2024 / Revised: March 23, 2025 / Accepted: May 28, 2025

Abstract—DonkeyCar is a DIY autonomous vehicle platform for exploring AI-driven self-driving technology. Using the DonkeyCar Unity simulator, models can be trained and tested in a risk-free environment before real-world deployment. This study examines the impact of training data and track complexity on model performance. Results show that increasing the number of training images from 1,346 to 5,351 reduced the error rate from 36% to 3%, thereby improving driving stability. Models trained on complex tracks adapted better but required longer training. However, challenges such as sensor noise, environmental uncertainties, and limitations in obstacle avoidance suggest the need for data augmentation, reinforcement learning, and multi-sensor fusion to enhance model robustness. These findings contribute to optimizing low-cost, vision-based autonomous driving systems.

Index Terms—Autonomous Driving, Deep Learning

I. INTRODUCTION

DonkeyCar is an open-source DIY platform designed for developing and training small-scale autonomous vehicles, primarily using RC cars as a base. It seamlessly integrates hardware and software, providing a user-friendly environment for experimenting with machine learning and autonomous driving technologies [1]. Whether for beginners or experienced developers, DonkeyCar offers a flexible framework for exploring autonomous vehicle development.

At the core of DonkeyCar is autonomous driving powered by machine learning? The development process begins with a user manually controlling the vehicle to collect driving data, including images and corresponding steering/acceleration commands. This dataset is then used to train a neural network model, which learns to predict the vehicle's control actions in unseen environments, ultimately enabling autonomous navigation [2].

Deep learning, a subfield of machine learning, relies on artificial neural networks to simulate the human brain's cognitive processes. It is particularly effective in handling unstructured data such as images and speech. Through multi-layer neural networks (deep networks), computers can automatically extract meaningful patterns and features from large datasets. In the DonkeyCar project [3], deep learning plays a crucial role in visual perception and decision-making. The vehicle captures road images through an onboard camera, which are then fed into a trained neural network. The model processes these images to determine appropriate steering and acceleration commands, facilitating autonomous driving.

Currently, most DonkeyCar projects rely primarily on cameras rather than additional sensors for three key reasons:

A. Simple and Efficient Visual Perception

Cameras provide a straightforward and effective means of capturing detailed environmental information, offering a direct representation of the car's surroundings. Visual perception is widely used in autonomous driving for its ability to detect lane markings, road signs, obstacles, and curves. Compared to other sensors, camera data is more intuitive and integrates seamlessly with deep learning models for real-time decision-making.

B. Low Cost and Accessibility

Cameras are cost-effective and widely available, making them an economical choice for DIY projects. Compared to expensive LIDAR or high-precision radar sensors, cameras provide a budget-friendly alternative that is familiar and easy to configure. Additionally, hardware platforms such as Raspberry Pi natively support camera modules, further simplifying system setup and reducing overall development costs [4].

C. Optimized for Deep Learning Models

The DonkeyCar platform heavily relies on deep learning models, particularly Convolutional Neural Networks (CNNs), which excel at processing visual

data. Cameras naturally complement CNN-based architectures, allowing for efficient end-to-end learning of driving behaviors. This approach streamlines the development process, enabling users to focus on training and optimizing image-based models without the added complexity of fusing data from multiple sensors [5].

D. Research Objective

This study aims to evaluate the impact of training data size and track complexity on the performance of vision-based autonomous driving models using DonkeyCar. By analyzing model accuracy, error rates, and generalization capabilities in different simulated environments, this work provides insights into optimizing low-cost, camera-based self-driving platforms. The findings contribute to improving autonomous navigation in real-world scenarios and advancing the development of AI-driven vehicle control systems.

II. RESEARCH OBJECTIVE

The primary objective of this research is to develop an autonomous vehicle following system based on the DonkeyCar platform. Specifically, this study aims to enhance vehicle

Perception and decision-making using deep learning-based vision systems. By leveraging Convolutional Neural Networks (CNNs), the proposed system will enable a trailing vehicle to detect and follow a leading vehicle in real-time, without relying on additional sensors such as LiDAR or radar.

To achieve this, the study will focus on the following key objectives:

(1) Data Collection and Model Training:

Develop a dataset by capturing real-time images from the onboard camera while manually driving the leading vehicle.

Train a deep learning model capable of recognizing the leading vehicle and estimating its position relative to the trailing vehicle.

(2) Autonomous Following Algorithm:

Implement a real-time object detection mechanism to identify the leading vehicle using CNN-based models.

Design a control strategy that allows the trailing vehicle to adjust its speed and steering angle based on the detected position of the leading vehicle.

(3) Simulation and Real-World Validation:

Test the proposed system in the DonkeyCar simulator to evaluate its performance in a controlled virtual environment.

Deploy the model on a physical DonkeyCar and conduct real-world experiments to assess robustness and adaptability under varying conditions.

The expected outcome of this research is a vision-based vehicle following system that operates

effectively in both simulated and real-world environments. The findings will help advance low-cost, camera-based autonomous driving solutions and provide valuable insights for future developments in small-scale autonomous vehicles [6].

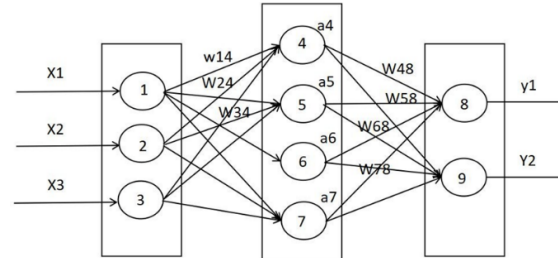


Fig. 1. Fully connected neural network (DNN)

III. RELATED WORKS

Autonomous driving has been a highly active research area, leveraging various Artificial Intelligence (AI) techniques to enhance perception, decision-making, and control. In particular, deep learning and Reinforcement Learning (RL) have demonstrated remarkable capabilities in enabling self-driving vehicles to navigate complex environments with minimal human intervention. In this section, we review the role of deep learning in autonomous driving, the application of reinforcement learning for adaptation and control, and the training workflow within DonkeyCar simulations.

A. Perception in Autonomous Driving

Perception is a critical component of autonomous driving, enabling vehicles to sense and interpret their surroundings. Deep learning models, particularly Convolutional Neural Networks (CNNs), have become the cornerstone of object detection, lane recognition, and traffic signal interpretation [7]. CNNs extract hierarchical spatial features from images, allowing vehicles to distinguish pedestrians, traffic lights, and other road elements.

Advanced object detection architectures, such as YOLO (You Only Look Once) and Faster R-CNN, have significantly improved real-time detection performance, making them suitable for embedded systems in autonomous vehicles [8]. These models help identify obstacles and make split-second navigation decisions, a crucial aspect of real-world deployment.

B. Deep Learning for Control and Decision Making

Deep learning is not only essential for perception but also plays a significant role in vehicle control. End-to-end learning approaches, where Deep Neural Networks (DNNs) map raw sensor inputs directly to steering commands, have shown promising results [9]. For instance, NVIDIA's DAVE-2 system trained a deep network using human driving data, successfully replicating human-like driving behaviors [10].

However, end-to-end learning often suffers from limited interpretability and robustness issues, making hybrid approaches more favorable. A combination of deep learning with traditional rule-based or model-based planning techniques provides a more reliable and scalable framework for autonomous navigation.

C. Reinforcement Learning for Adaptive Driving

Reinforcement Learning (RL) has emerged as a powerful paradigm for training self-driving cars in dynamic and uncertain environments. Unlike supervised learning, where models learn from labeled data, RL enables agents to learn optimal policies through trial and error, receiving rewards for desirable actions [11].

Proximal Policy Optimization (PPO) and Deep Q-Networks (DQN) are widely used RL algorithms for autonomous driving [12]. These methods allow self-driving cars to refine their control strategies, adapt to varying road conditions, and optimize fuel efficiency.

D. DonkeyCar Training Workflow

DonkeyCar provides a lightweight and accessible platform for training autonomous driving models in simulation before deploying them on real-world robotic vehicles. The training pipeline follows a structured workflow:

(1) Data Collection:

The DonkeyCar vehicle is manually driven in a simulated environment to collect training images and sensor data.

(2) Preprocessing and Augmentation:

The collected data is cleaned and augmented to enhance model robustness. Techniques such as brightness adjustment, cropping, and flipping help the model generalize better.

(3) Model Training:

A CNN-based model is trained to map images to steering and throttle commands. This model learns to navigate autonomously by imitating human driving patterns.

(4) Evaluation and Fine-Tuning:

The trained model is tested in the simulator to evaluate its performance. If necessary, additional training data is collected, and hyperparameters are tuned to improve accuracy.

(5) Deployment:

Once the model achieves satisfactory performance, it is deployed on a physical DonkeyCar, allowing it to navigate autonomously in real-world scenarios.

Empirical results indicate that increasing the number of training samples significantly enhances model accuracy and driving stability. As shown in Table I, a model trained on 5,351 images exhibits a

considerably lower error rate than one trained on only 1,346 images [13].

TABLE I
MODEL COMPARISON

Model	Training Images	Collisions	Error Rate
Model 1	1,346	36	36%
Model 2	5,351	3	3%

E. Challenges and Future Directions

Despite significant advancements, DonkeyCar-based autonomous driving still faces several challenges:

- **Limited Computational Power:**

The embedded systems used in DonkeyCar have lower processing capabilities than those of full-scale autonomous vehicles, limiting the complexity of deployable models.

- **Generalization to Real-World Scenarios:**

While models trained in simulation perform well within controlled environments, their transferability to real-world conditions remains an open challenge.

- **Automated Labeling and Data Augmentation:**

Current training methodologies rely heavily on manually collected data. Future work could explore automated labeling techniques and synthetic data generation to improve scalability.

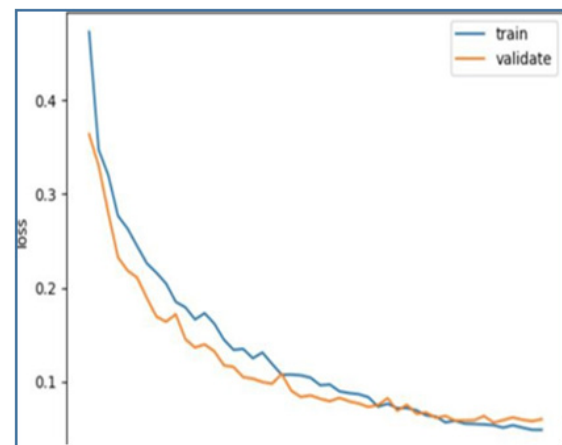


Fig. 2. Model loss picture

As illustrated in the figure above, the blue line represents the model's training curve, while the validation curve serves as a comparison. The close alignment between the two curves suggests that the model generalizes well. The DonkeyCar simulator was used for training [14], and, owing to the extensive dataset collected, the model exhibits strong autonomous driving capabilities with minimal lane deviations.

part	max	min	avg	50%	90%	99%	99.9%
DonkeyGymEnv	0.89	0.01	0.03	0.03	0.04	0.08	0.19
LocalWebController	3.45	0.01	0.09	0.04	0.18	0.75	2.48
Pipe	0.32	0.00	0.01	0.01	0.02	0.04	0.07
ExplodeDict	0.19	0.00	0.01	0.01	0.01	0.03	0.07
ThrottleFilter	0.36	0.00	0.01	0.01	0.01	0.03	0.07
UserPilotCondition	0.38	0.00	0.01	0.01	0.02	0.04	0.09
RecordTracker	4.91	0.00	0.02	0.01	0.01	0.24	1.00
AltLaunch	0.41	0.00	0.01	0.01	0.01	0.03	0.08
DriveMode	0.30	0.00	0.01	0.01	0.02	0.04	0.10
ToggleRecording	6.28	0.00	0.10	0.01	0.34	1.67	4.07
TubWriter	43.94	0.58	4.61	4.48	7.00	13.78	40.29

Fig. 3. Training model diagram

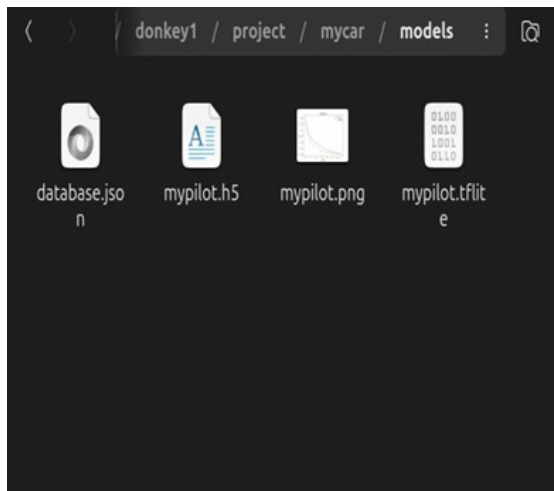


Fig. 4. Training model diagram

After training with DonkeyCar, a series of essential files are generated, including mypilot.h5, database.json, mypilot.png, and mypilot.tflite. These files collectively form the core data of the model, ensuring its usability, reproducibility, and smooth deployment.

- **mypilot.h5 File:** This is the primary output file from DonkeyCar training, which stores the neural network's trained weights. It records how the model learns to associate input images from the car's camera with user-provided control signals (steering and throttle). For example, when enabling autonomous driving, DonkeyCar loads the mypilot.h5 file, allowing the vehicle to drive autonomously based on the training data [15].

- **database.json File:** This file stores metadata about the trained model, such as network architecture, training parameters, vehicle information, speed, model name, and local control records. It plays a crucial role in reloading the model, inspecting its configuration, or further training. Additionally, this file describes

the neural network architecture used for training, including the number of layers, activation functions, and other custom components, ensuring that the model structure can be accurately recreated for inference or retraining [16].

- **mypilot.tflite File:**

This is a TensorFlow Lite model optimized for lightweight inference. It is particularly useful for deployment on resource-constrained devices like Raspberry Pi, enabling efficient execution of the trained model on low-power hardware.

- **mypilot.png File:**

This file typically contains visualized data generated during training, such as the model's loss curve or driving trajectory, helping analyze and optimize training performance.

These files collectively form the complete model output after DonkeyCar training, enabling easy reloading, inspection, and efficient deployment across different devices, ensuring the reproducibility and reliability of the trained model [17].

```

/home/tyangyang/miniconda3/envs/donkey/lib/python3.11/site-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via 'model.save()'. This file format is considered legacy. We recommend using instead the native Keras format, e.g. 'model.save('my_model.keras')'.
  saving_api.save_model(
12/12 [=====] - 12s 827ms/step - loss: 0.3681 - n_outputs0_loss: 0.3142 - n_outputs1_loss: 0.0539 - val_loss: 0.3484 - val_n_outputs0_loss: 0.3098 - val_n_outputs1_loss: 0.0387
Epoch 2/100
1/12 [=>.....] - ETA: 6s - loss: 0.3105 - n_outputs0_loss: 0.2683 - n_outputs1_loss: 0.0330
2/12 [====>...] - ETA: 6s - loss: 0.3283 - n_outputs0_loss: 0.3330 - n_outputs1_loss: 0.3276
3/12 [=====>...] - ETA: 6s - loss: 0.3330 - n_outputs0_loss: 0.3276 - n_outputs1_loss: 0.3269
4/12 [=====] - ETA: 5s - loss: 0.3276 - n_outputs0_loss: 0.3313 - n_outputs1_loss: 0.3330
5/12 [=====] - ETA: 5s - loss: 0.3269 - n_outputs0_loss: 0.3330 - n_outputs1_loss: 0.3330
6/12 [=====] - ETA: 4s - loss: 0.3313 - n_outputs0_loss: 0.3330 - n_outputs1_loss: 0.3330
7/12 [=====] - ETA: 3s - loss: 0.3330 - n_outputs0_loss: 0.3330 - n_outputs1_loss: 0.3330

```

Fig. 5. Train Model

Formula:

$$MSE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

Description: Mean Squared Error (MSE) is the most commonly used loss function in DonkeyCar for regression tasks. It measures the average squared difference between predicted and actual values, penalizing larger errors more heavily [18].

Use Case: MSE is ideal for predicting continuous values, such as steering angles and throttle, to ensure smooth and precise vehicle control.

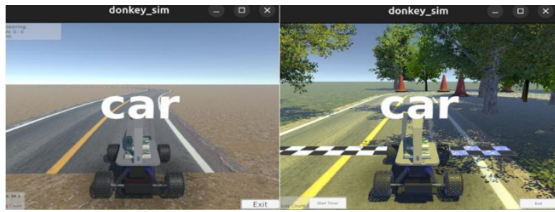


Fig. 6. Comparison of two tracks

Fig. 6 illustrates two distinct tracks within the simulator: **generated_track** and **generated_road**. This experiment aims to analyze differences in model training across the two tracks, assess the impact of training data on model performance, and explore optimization strategies to enhance the trained model's effectiveness for autonomous driving.

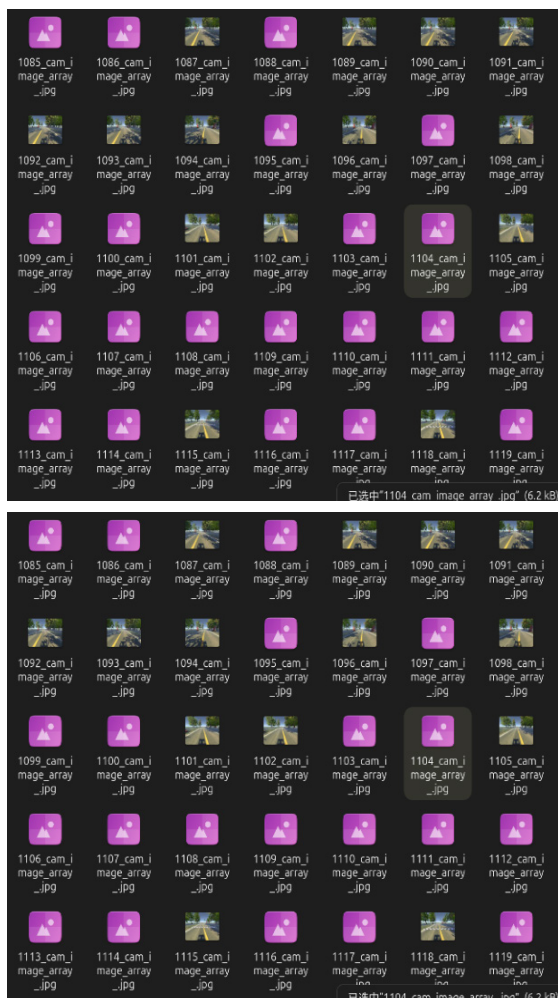


Fig. 7. Generated_road data

The two images above show the second map used for training, **generated_track**. Initially, 1,346 images were used, and the results showed 36 collisions, with the model unable to complete a full lap. This indicates insufficient data, shorter training time, and a more complex map with more obstacles.

As shown in Fig. 10, when the data increased to 5,351 images, the model's performance improved significantly. It was able to complete autonomous driving on the main road, and the error rate dropped from 36% to 3%.

From these two map examples, it is clear that the model's performance is closely related to the amount of training data. During training, incorrect input data can affect the model, and with less data, errors are more likely to be included in the model. As the data set grows, the error rate decreases significantly [19]. The three images above show the automatic driving scenario generated by the model. The vehicle cannot autonomously avoid obstacles, possibly due to unclear information about certain road sections during training, resulting in erroneous data. Additionally, the small data set prevents the model from independently identifying obstacles [20].

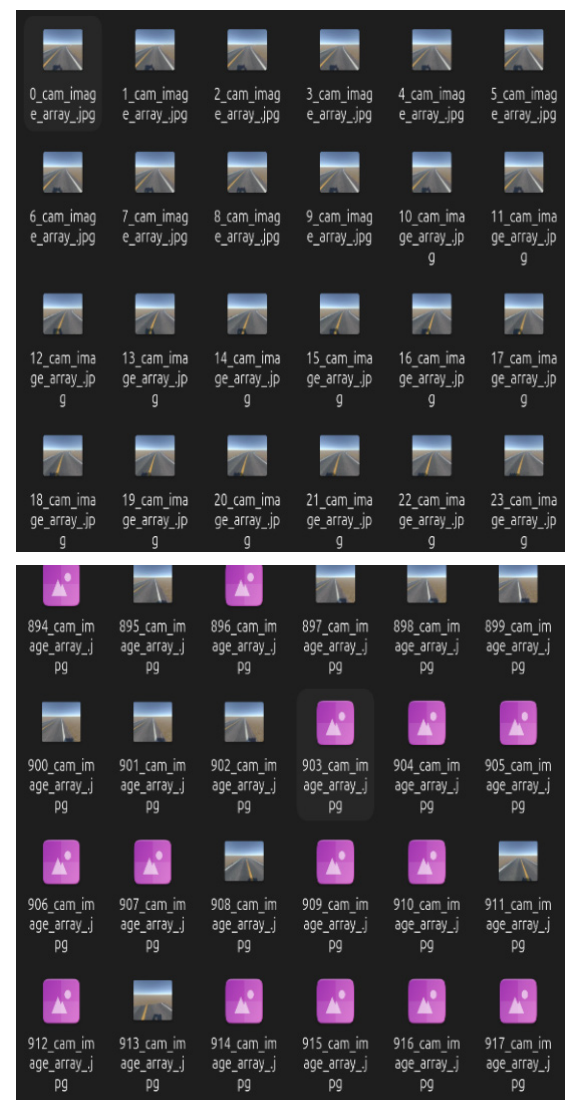


Fig. 8. Generated track data

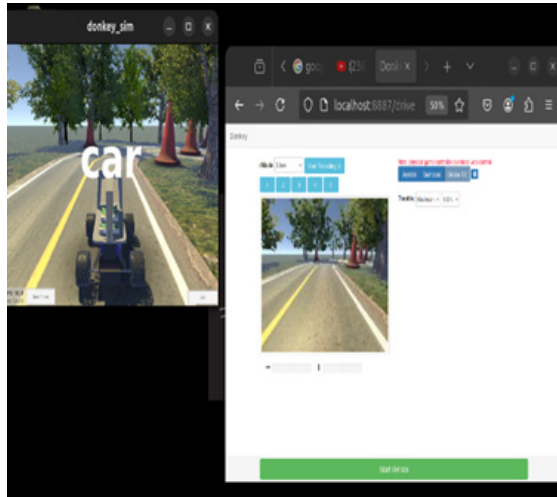


Fig. 9. Autonomous tracking

The experimental platform we built is based on the Donkey Car open-source autonomous driving platform, which implements a layered design.

We also built the model's training and inference environments. Table I presents a detailed configuration of the software and hardware utilized for the experiments in this work.

TABLE II
SOFTWARE AND HARDWARE CONFIGURATIONS

Category	Details
CPU	Intel Core i7-9700KF @ 3.60GHz
GPU	NVIDIA GeForce RTX 2060
RAM	16GB DDR4 @ 3200MHz
OS	Linux
Language	Python 3.11
Framework	TensorFlow-GPU 2.2
Libraries	OpenCV 4.5, NumPy 1.21

IV. RESULT

The experimental results demonstrate that increasing the training data size and incorporating diverse driving environments significantly enhance the performance of autonomous driving models. Initially, with only 1,346 training images, the model encountered 36 collisions and was unable to complete a full lap. However, after expanding the dataset to 5,351 images, the model achieved stable autonomous navigation, with the error rate decreasing from 36% to 3%.

Additionally, when tested on two different tracks (generated_track and generated_road), models trained on simpler tracks exhibited smoother driving behavior, whereas those trained on more complex tracks required additional training epochs to achieve

similar accuracy. These findings highlight the crucial role of data diversity and track complexity in training robust self-driving models, suggesting that a more varied dataset can significantly improve generalization and driving stability.

V. DISCUSSION

The results highlight several key factors affecting the performance of DonkeyCar-based autonomous driving models:

Training Data Size: A larger dataset leads to lower error rates and improved driving stability. Future research should explore data augmentation techniques such as brightness adjustment, flipping, and synthetic image generation.

Obstacle Avoidance Limitations: The model exhibited limitations in detecting and avoiding obstacles. Future improvements could integrate LiDAR or ultrasonic sensors for better environmental awareness.

Unlike general deep learning tasks, autonomous driving presents unique challenges, particularly in handling edge cases such as unexpected pedestrian crossings or sudden lane changes. To address this, loss functions such as Huber Loss are often preferred over standard MSE, as they reduce sensitivity to outliers. Additionally, data augmentation techniques, including brightness variation, motion-blur simulation, and adversarial occlusions, can improve model robustness across varying environmental conditions.

Real-World Deployment: While the model performs well in simulation, its transferability to real-world conditions remains a challenge. Future work should explore domain adaptation techniques to bridge the gap between simulation and reality.

Reinforcement Learning Enhancements: Instead of purely supervised learning, integrating Deep Q-Networks (DQN) or Proximal Policy Optimization (PPO) could allow the model to learn adaptive driving strategies through trial and error.

VI. CONCLUSION

This study successfully developed an autonomous driving model using DonkeyCar, demonstrating lane following and partial obstacle avoidance. The results emphasize the importance of data quality, road complexity, and hardware efficiency in improving the performance of self-driving AI models.

Future work will focus on expanding the dataset, incorporating additional sensors, and improving obstacle detection algorithms. With these advancements, the model can transition from simulation to real-world deployment, paving the way for future innovations in AI-driven autonomous vehicles.

ACKNOWLEDGMENT

The first author received scholarship support from CP ALL to conduct this research at PIM. The last author is the corresponding author.

REFERENCES

- [1] J. Janai, F. G. Güney, A. Behl, and A. Geige, "Computer vision for autonomous vehicles: Problems, datasets, and state-of-the-art," *Found. Trends Comput. Graph. Vis.*, vol. 12, no. 13, pp. 1-232, Apr. 2017.
- [2] H. Abraham, L. Chaiwoo, S. Brady, and C. Fitzgerald, "Autonomous vehicles, trust, and driving alternatives: A survey of consumer preferences," in *Proc. Transp. Res. Board 96th Annu. Meeting*, 2017, pp. 1-16.
- [3] R. S. Ferreira, "Towards safety monitoring of ML-based perception tasks of autonomous systems," in *Proc. IEEE Int. Symp. Softw. Reliab. Eng. Workshops (ISSREW)*, 2020, pp. 135-138.
- [4] D. Fang et al., "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1341-1360, Mar. 2021.
- [5] A. Ramisa, F. Yan, F. Moreno-Noguer, and K. Mikolajczyk, "Breaking news: Article annotation by image and text processing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 12, pp. 2487-2500, Dec. 2016.
- [6] D. Shukai and J. Qu, "Research on multi-tasking smart cars based on autonomous driving systems," *SN Comput. Sci.*, vol. 4, no. 3, p. 292, Mar. 2023.
- [7] Z. Jing, P. Li, B. Wu, S. Yuan, and Y. Chen, "An adaptive focal loss function based on transfer learning for few-shot radar signal intra-pulse modulation classification," *Remote Sens.*, vol. 14, no. 8, p. 1950, Aug. 2022.
- [8] H. Lai, L. Chen, W. Liu, Z. Yan, and S. Ye, "STC-YOLO: Small object detection network for traffic signs in complex environments," *Sensors*, vol. 23, no. 11, Jun. 2023.
- [9] M. Andrychowicz et al., "Learning dexterous in-hand manipulation," *Int. J. Robot. Res.*, vol. 39, no. 1, pp. 3-20. <https://doi.org/10.1177/0278364919887447>
- [10] J. Tan et al., "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv*, 2018. [Online]. Available: <https://arxiv.org/abs/1804.10332> [Accessed: Sep. 26, 2024].
- [11] A. Kendall et al., "Learning to drive in a day," *arXiv*, 2018. [Online]. Available: <https://arxiv.org/abs/1807.00412> [Accessed: Sep. 26, 2024].
- [12] D. Dörr, D. Grabenigier, and F. Gauterin, "Online driving style recognition using fuzzy logic," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, 2014, pp. 1021-1026.
- [13] M. Bojarski et al., "End-to-end learning for self-driving cars," *arXiv*, 2016. [Online]. Available: <https://arxiv.org/abs/1604.07316> [Accessed: Sep. 26, 2024].
- [14] Y. Li and J. Qu, "MFPE: A loss function based on multi-task autonomous driving," *ECTI Trans. Comput. Inf. Technol.*, vol. 16, no. 4, pp. 393-409, Oct. 2022.
- [15] V. Ghodrati et al., "MR image reconstruction using deep learning: Evaluation of network structure and loss functions," *Quant. Imaging Med. Surg.*, vol. 9, no. 9, pp. 1516-1527, Sep. 2019.
- [16] M. Patel and H. Elgazzar, "Classification of road objects using convolutional neural networks," in *Proc. IEEE Consum. Commun. Netw. Conf. (CCWC)*, 2023, pp. 326-332.
- [17] M. Bechtel, E. McElhiney, and H. Yun, "DeepPicar: A low-cost deep neural network-based autonomous car," in *Proc. IEEE Int. Conf. Embedded Real-Time Comput. Syst. Appl. (RTCSA)*, 2018, pp. 11-21.
- [18] J. Maanpää et al., "Multimodal end-to-end learning for autonomous steering in adverse road and weather conditions," in *Proc. Int. Conf. Pattern Recognit. (ICPR)*, 2021, pp. 699-706.
- [19] S. Azam et al., "N2C: Neural network controller design using behavioral cloning," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4744-4756, Jul. 2021.
- [20] A. Loquercio, A. I. Maqueda, C. R. del-Blanco, and D. Scaramuzza, "DroNet: Learning to fly by driving," in *Proc. IEEE Robot. Autom. Lett.*, 2018, pp. 1088-1095.



Reinforcement Learning.

Yangyang Li is currently pursuing a Master's degree at the Faculty of Engineering, Panyapiwat Institute of Management, Thailand. We are engaged in developing secure and reliable autonomous driving systems leveraging Deep



Jian Qu is a full-time lecturer at the Faculty of Engineering and Technology, Panyapiwat Institute of Management. He received a Ph.D. with an Outstanding Performance award from the Japan Advanced Institute of Science and Technology, Japan, in 2013. He received a B.B.A with Summa Cum Laude honors from the Institute of International Studies of Ramkhamhaeng University, Thailand, in 2006, and M.S.I.T from Sirindhorn International Institute of Technology, Thammasat University, Thailand, in 2010. His research interests are natural language processing, intelligent algorithms, machine learning, machine translation, information retrieval, and image processing.